TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TERO PYYLAMPI

REAALIAIKAISEN MONINPELIJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS ANDROID-KÄYTTÖJÄRJESTELMÄLLE

Diplomityö

Tarkastaja: professori Kari Systä
Jätetty tarkastettavaksi: 24.11.2015
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston kokouksessa
3. kesäkuuta 2015

TIIVISTELMÄ

Pyylampi, Tero: Reaaliaikaisen moninpelijärjestelmän suunnittelu ja toteutus Android-käyttöjärjestelmälle
Tampereen teknillinen yliopisto
Diplomityö, 64 sivua
Marraskuu 2015
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Kari Systä
Avainsanat: android, vertaisverkot, p2p, moninpelit

Reaaliaikaisesti muiden pelaajien kanssa pelattavat moninpelit ovat yleistyneet mobiilipelimaailmassa merkittävästi. Tähän kehitykseen on osaltaan vaikuttanut niin mobiililaitteiden ja -käyttöjärjestelmien kuin langattomien verkkoyhteyksienkin tekninen kehitys, mutta suurin merkitys lienee kuitenkin kuluttajien eli pelaajien kasvaneilla tarpeilla. Ovathan internetin yli pelattavat verkkopelit olleet arkipäivää PC- ja konsolimaailmassa jo lähes kahden vuosikymmenen ajan.

Olemassaolevia järjestelmiä moninpelien toteuttamiseksi mobiililaitteille on saatavilla toistaiseksi varsin niukasti, ja niiden käyttöön saattaa liittyä joitakin rajoituksia esimerkiksi käyttäjämäärien tai saatavuuden suhteen. Lisäksi järjestelmien perustuessa suljettuun lähdekoodiin, olemassaolevien virheiden korjaaminen tai toimintojen lisääminen on käytännössä mahdotonta.

Tässä diplomityössä tutkitaan Hyperkani Oy -yrityksen tällä hetkellä hyödyntämän Google Play Games Services SDK -moninpelirajapinnan korvaamista itse toteutetulla järjestelmällä. Ratkaisun toiminta perustuu vertaisverkkoihin sekä avoimen lähdekoodin Libnice-kirjastoon, eikä se ole riippuvainen kolmannen osapuolen rajoitetusti saatavilla olevista komponenteista. Tämä mahdollistaa järjestelmän hyödyntämisen myös sellaisilla maantieteellisillä alueilla, joissa Google Play -palvelun toiminnot eivät ole saatavilla.

Vaikka järjestelmä toimiikin suunnitellusti, sitä ei kuitenkaan toistaiseksi otettu yrityksessä käyttöön. Testausprosessin aikana havaittiin suuren osan mobiiliverkkojen käyttäjistä sijaitsevan varsin rajoittavien osoitteenmuunnosten tai palomuurien takana, jolloin suorien vertaisverkkoyhteyksien muodostaminen ei ollut sellaisenaan mahdollista. Ratkaisu tukee erillisen relay-palvelimen käyttämistä tällaisten tahojen väliseen yhteydenmuodostukseen, mutta koska lähes kaikki yhteydet joudutaan muodostamaan tämän välityspalvelimen kautta, ei ratkaisu kelpaa käytössä olevan järjestelmän korvaajaksi ilman kehittyneempää yhteydenmuodostusalgoritmia.

ABSTRACT

Pyylampi, Tero: Design and implementation of a real-time multiplayer system for Android operating systems
Tampere University of Technology
Master of Science Thesis, 64 pages
November 2015
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Professor Kari Systä
Keywords: android, peer-to-peer networks, multiplayer games

Real-time multiplayer games have taken a substantial stand in the mobile world during the last few years. Partially this has been made possible by the constantly advancing technology of mobile devices and -networks, but still most of the credit should be given to the increasing demand amongst the players. After all, online games have already been available for both PC and console platforms for almost two decades.

Available frameworks especially purposed for mobile online game development are still rather uncommon, and some of them also have various restrictions. Additionally, as the frameworks are typically closed source and/or commercial, it is either difficult or impractical to tailor the frameworks to match the developers needs.

This thesis was made for the company called Hyperkani Ltd. and it investigates the replacement of the currently used mobile multiplayer framework Google Play Games Services with a custom-made, more suitable solution. This new solution is based on peer-to-peer networks and open source libraries, and it does not depend on any closed source components or proprietary products. This allows the solution to be used even on locations where Google Play -services are inaccessible.

Although the system works and performs as intended, the testing process revealed that most of the mobile network users are either behind restrictive network address translations or firewalls. This effectively prevents the use of direct connections between peers. In such cases, the system forms connections using an intermediate relay-server designed to transmit data between peers. Unfortunately, as this appears to work as a default configuration instead of being a fallback solution, further development is required before the system can be adopted to production in Hyperkani Ltd.

ALKUSANAT

Olen aina ollut varsin aktiivinen eri pelityyppien harrastaja. Siksi olinkin mielissäni, kun sain tehdä diplomityöni näin lähellä omaa mukavuusaluettani. Mielenkiinnosta ja harrastuneisuudesta huolimatta työn tekninen osio osoittautui aluksi varsin haastavaksi, mutta sitäkin palkitsevammalta tuntuu saada tämä työ lopulta valmiiksi. Toivottavasti työn sisällöstä on iloa myös muille. Mahdolliset työhön liittyvät kysymykset voi esittää alla olevaan sähköpostiosoitteeseen.

Suuri kiitos professori Kari Syställe diplomityöni ohjaamisesta, kommenteista, palautteesta, sekä tarkastamisesta. Kiitos työnantajalleni Hyperkani Oy:lle sekä erityisesti esimiehelleni Mikko Vartialalle jatkuvasta opastamisesta, työni ajoittaisista tarkastuksista ja työn säännöllisestä eteenpäin potkimisesta. Kiitos myös avopuolisolleni Saaralle pilkkujen viilaamisesta, tekijän avautumisten kestämisestä sekä muusta työhön enemmän tai vähemmän liittyvästä jatkuvasta tuesta.

Edellämainittujen lisäksi haluan kiittää äitiäni säännöllisestä ruokahuollosta, muuta perhettäni taustapiruilusta ja kavereitani maan pinnalle palauttamisesta erilaisten pelien ja illanistujaisten merkeissä.

Tampereella, 24.11.2015

Tero Pyylampi

t.a.pyylampi@gmail.com

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

Android	Linux-käyttöjärjestelmäytimeen perustuva mobiililaitteille suunniteltu ohjelmistopino, joka sisältää käyttöjärjestelmän.
GLib	Kokoelma C-kielellä kirjoitettuja matalan tason järjestelmäkirjastoja.
IAP	In-App Purchase, pelin tai ohjelman sisäinen ostotapahtuma.
ICE	Interactive Connectivity Establishment, kokoelma erilaisia protokollia suoran yhteyden muodostamiseksi tahojen välille.
iOS	Applen kehittämä käyttöjärjestelmä, joka on käytössä useissa Applen tuoteperheissä.
JNI	Java Native Interface, mahdollistaa virtuaalikoneen ja natiivikoodilla toteutettujen osioiden välisen kommunikoinnin.
NAT	Osoitteenmuunnostekniikka, jossa julkisesti liikennöityjä IP-osoitteita säästetään keräämällä useampia osoitteita yhden julkisen osoitteen taakse.
NDK	Native Development Kit, kokoelma ohjelmistonkehitystyökaluja, joiden avulla saadaan luotua ohjelmia kohdejärjestelmälle natiivikoodilla, kuten esimerkiksi C++:lla.
P2P	Vertaisverkko (Peer-to-Peer), verkko, jossa jokainen osapuoli toimii sekä asiakkaana että palvelimena muille verkon jäsenille.
SDK	Valikoima kehitystyökaluja ja/tai kirjastoja jonkun tietyn toiminnallisuuden toteuttamiseen.

STUN	Protokolla (Simple Traversal of UDP through NATs), jota käytetään selvittämään onko jokin verkon taho osoitteenmuunnoksen takana sekä erityistapauksissa myös ohjaamaan osoitteenmuunnoksen suorittavaan laitteeseen saapuvia UDP-paketteja tietylle asiakkaalle. Käytetään vertaisverkon muodostamiseen.
TCP	Tietoliikenneprotokolla (Transmission Control Protocol), joka määrittelee luotettavan yhteyden kahden verkotetun laitteen välille.
TURN	Protokolla (Traversal Using Relay NAT), jossa vertaisverkko muodostetaan erillisen julkisen välittäjäpalvelimen kautta.
UDP	Yhteydetön protokolla (User Datagram Protocol), joka eroaa TCP:stä siten, että pakettien perillemenoa ei taata eikä yhteyksiä avattaessa tai suljettaessa ole erillisiä kättelymekanismeja.
UPnP	Universal Plug and Play, kokoelma verkkoprotokollia, jotka mahdollistavat samassa verkossa sijaitsevien laitteiden keskenäisen yhteyden muodostamisen automaattisesti.
VPN	Virtuaalinen erillisverkko (Virtual Private Network), menetelmä, jossa kaksi erillistä verkkoa voidaan yhdistetään julkisen internetin yli.
WebRTC	Ohjelmointirajapinta, joka mahdollistaa selainten väliset reaaliaikaiset vertaisverkkoyhteydet ilman selainlaajennoksia.
XMPP	Pikaviestintään ja läsnäolon seurantaan kehitetty protokolla.

1. JOHDANTO

Hyperkani Oy on mobiilipeleihin keskittyvä mikroyritys. Yrityksen ansaintamalli perustuu pelaamisen ohella pelaajille näytettäviin mainoksiin sekä pelien sisäisiin ostoihin (IAP). Mainoksia näyttävät komponentit pyrkivät käyttäjän sijainnin sekä mahdollisten muiden käyttäjästä saatavien tietojen perusteella kohdentamaan juuri kyseiselle käyttäjälle soveltuvimpia joko bannerin tai koko ruudun kokoisia mainoksia, kun taas pelien sisäisissä ostoissa käyttäjän on mahdollista ostaa esimerkiksi kultaa tai vastaavaa käytössä olevaa pelivaluuttaa oikealla rahalla. Hyperkani Oy:n nykyiset pääasialliset kohdealustat ovat Android sekä iOS -käyttöjärjestelmät, mutta myös muille mobiilialustoille on julkaistu pelejä.

Mobiilidatayhteyksien (3G, 4G, WiFi) kehittyessä pelaaminen on mahdollista myös muiden eri puolilla maailmaa sijaitsevien pelaajien kanssa. Olennaisena osana verkkopelaamiseen kuuluu tiedon, kuten esimerkiksi pelitilanteen, vaihtaminen muiden pelaajien kesken. Pelin luonne voi asettaa tietojen vaihtamiselle joitakin rajoituksia ajan, tietoturvan, tiedon eheyden tai muiden ominaisuuksien suhteen. Yksinkertaisessa monen pelaajan korttipelissä voi esimerkiksi olla oleellista, että pelaajien saamat kortit arvotaan erillisellä pelipalvelimella, joka vastaa siitä, ettei sama kortti voi esiintyä kahdella eri pelaajalla samanaikaisesti.

Toisaalta taas esimerkiksi reaaliaikaisessa toimintapelissä aikakriittisyys voi olla paljon tärkeämpi tekijä pelin sujuvuuden kannalta: pelimaailmaan kohdistuvien muutosten tulee päivittyä kaikille pelaajille mahdollisimman nopeasti. Tällaisissa aikakriittisissä sovelluksissa kaiken tiedon välittäminen erillisen keskuspalvelimen kautta muille pelaajille voi olla liian hidasta. Lisäksi pelaajamäärien kasvaessa tämä voi myös kuormittaa keskuspalvelinta tarpeettoman paljon.

Mikäli aiemmin mainittu keskuspalvelin ei ole välttämätön esimerkiksi tiedon eheyden tai oikeellisuuden tarkasteluun, voidaan pelaajien välille muodostaa myös suora yhteys, jossa jokainen osapuoli toimii sekä asiakkaana että palvelimena muille verkon jäsenille. Tällaista eri asiakkaiden välistä yhteyttä kutsutaan vertaisverkoksi (Peer-to-Peer, P2P). Mobiililaitteiden väliset vertaisverkot eivät ole vielä toistaiseksi kovin yleisiä mobiiliverkkojen asettamista rajoitteista johtuen, mutta niiden käyttö-

aste kasvaa jatkuvasti.

Eräs vertaisverkkoihin perustuva moninpelikomponentti on Googlen Play Games Services SDK:n sisältämä ohjelmointirajapinta. Tämän komponentin avulla sovellus muodostaa ensin yhteyden kiinteään keskuspalvelimeen sekä kirjautumista että kanssapelaajien etsimistä (matchmaking) varten, jonka jälkeen löydetyt pelaajat muodostavat vertaisverkon keskenään. Komponentti on helppokäyttöinen ja sisältää kaikki tarvittavat perustoiminnot verkkopelien kehittämiseen, mutta sen lähdekoodi on suljettua, eivätkä komponentin tarjoamat ominaisuudet ole kovinkaan monipuolisia. Vaikka komponentti onkin saatavilla myös iOS-järjestelmille, se vaatii käyttäjiltään Google Play -tunnukset, jotka ovat Apple-tuotteiden käyttäjien keskuudessa varsin harvinaisia. Lisäksi pelaajamäärien kasvaessa tietyn rajan yli komponentin käyttö muuttuu maksulliseksi.

Play Games Services SDK on käytössä myös joissakin Hyperkani Oy:n toteuttamissa sovelluksissa. Aiemmin mainituista rajoitteista johtuen yrityksen intressinä on korvata tämä komponentti vaihtoehtoisella ratkaisulla, joka olisi monipuolisempi ja perustuisi avoimeen tai yrityksen omaan lähdekoodiin. Kriittisimmät ratkaisulta vaaditut ominaisuudet liittyvät suorituskykyyn, vasteaikoihin, sekä suurien käyttäjämäärien samanaikaiseen tukemiseen. Ratkaisun tulisi olla sellaisenaan käytettävissä sekä Android että iOS -käyttöjärjestelmillä.

Tässä työssä pyritään etsimään annettuihin kriteereihin sopivinta ratkaisua sekä käsitellään prosessin aikana hylättyjä ratkaisuehdotuksia. Löydetystä ratkaisusta esitellään prototyyppi sen käyttökelpoisuuden osoittamiseksi. Lisäksi tämän prototyypin suorituskykyä mitataan erilaisilla viivetesteillä sekä arvioidaan sen soveltuvuutta tarkoitukseensa myös muiden ominaisuuksien osalta. Mikäli prototyyppi osoittautuu toimivaksi ja käyttäkelpoiseksi ratkaisuksi, siitä tullaan kehittämään valmis toteutus.

Prototyypin rakentamiseen liittyvät haasteet muodostuvat pääosin kohdekäyttöjärjestelmien sekä matkapuhelinverkkojen asettamista rajoitteista. Toistaiseksi olemassa olevien vapaan lähdekoodin vertaisverkkokirjastojen mobiilijärjestelmätuki on varsin puutteellista ja lisäksi suurin osa mobiilidataverkkoja hyödyntävistä laitteista on osoitteenmuunnoksen takana. Prototyypin toiminta osoittaa kuitenkin ratkaisun olevan mahdollinen sekä toteutettavissa.

Työn sisältö jakautuu seuraavasti: johdannon jälkeisessä luvussa käsitellään Android-käyttöjärjestelmää sekä sen toimintaa niiltä osin kuin on katsottu tarpeelliseksi tä-

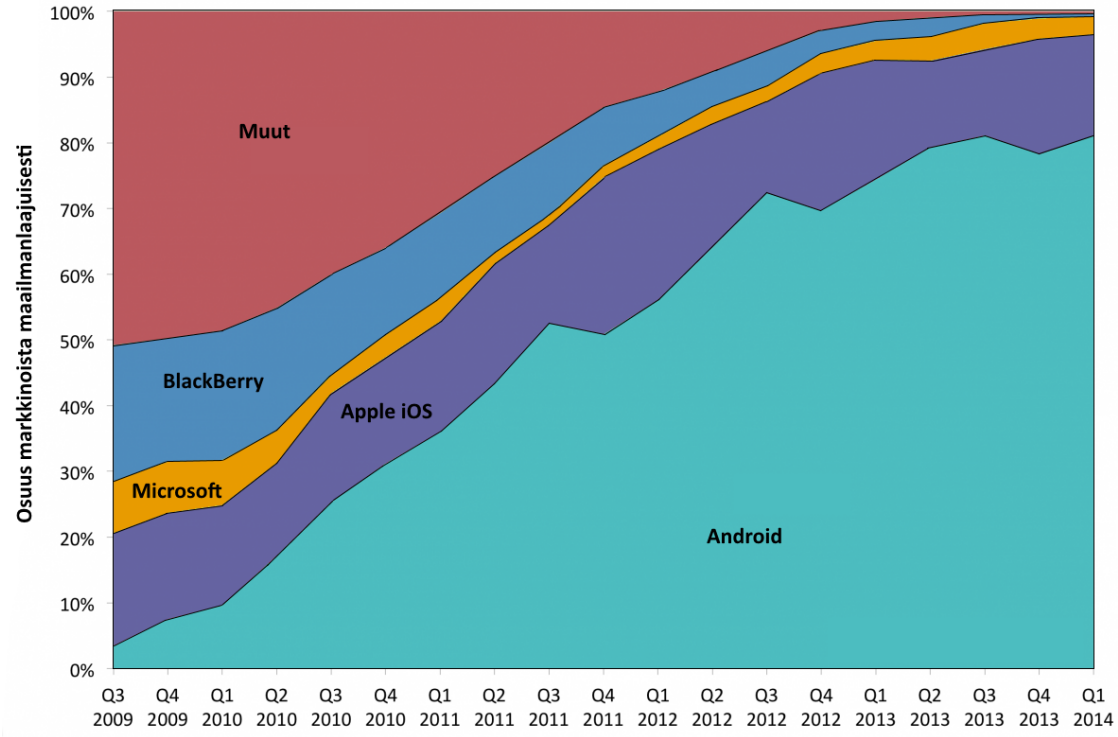
män työn ymmärtämiseksi. Kolmannessa luvussa keskitytään samaan vertaisverkkojen toimintaperiaatteiden osalta. Tämän jälkeen tarkastellaan käytössä olevaa järjestelmää, siinä esiintyviä puutteita sekä ongelman ratkaisulle asetettavia vaatimuksia. Luvussa viisi puolestaan esitellään ratkaisuvaatimukset toteuttava prototyyppi. Kuudennessa luvussa selvitetään komponentin käyttöönottoon liittyviä huomioita, kun taas seitsemännessä luvussa arvioidaan luotua prototyyppiä niin suorituskyvyn kuin muidenkin näkökulmien osalta. Kahdeksannessa luvussa käsitellään erilaisia prototyyppiin liittyviä kehityskohteita ja keskitytään erityisesti suorien vertaisverkkoyhteyksien muodostamiseen. Lopulta yhteenvedossa pohditaan ratkaisun sekä työn onnistumista eri näkökulmista. Työn lopussa on listattu työn tekemisen aikana käytetyt lähteet.

2. ANDROID PELIALUSTANA

Tässä luvussa käsitellään Android-käyttöjärjestelmän toimintaa ohjelmisto- sekä rautatasolla niiltä osin kuin on tarpeellista tämän työn ymmärtämiseksi.

2.1 Käyttöjärjestelmän historia ja suosio

Vaikka Androidista kuultiin ensimmäisen kerran julkisuudessa jo vuonna 2005 Googlen ostaessa omistusoikeuden pieneen startup-yritykseen nimeltä Android Inc, käyttöjärjestelmän 1.0 versio julkaistiin kuitenkin vasta syksyllä 2008 kuluttajien saataville [65, s. 2]. Järjestelmälle löytyi välittömästi oma käyttäjäkuntansa, ja kuten kuvasta 2.1 havaitaan, Android saavutti mobiilikäyttöjärjestelmien markkinajohtajuuden vuoden 2011 toisella neljänneksellä sen kasvun ollessa jatkuvassa nousussa julkaisustaan lähtien.



Kuva 2.1: *Eri mobiilijärjestelmien suosio vertailussa [13]*

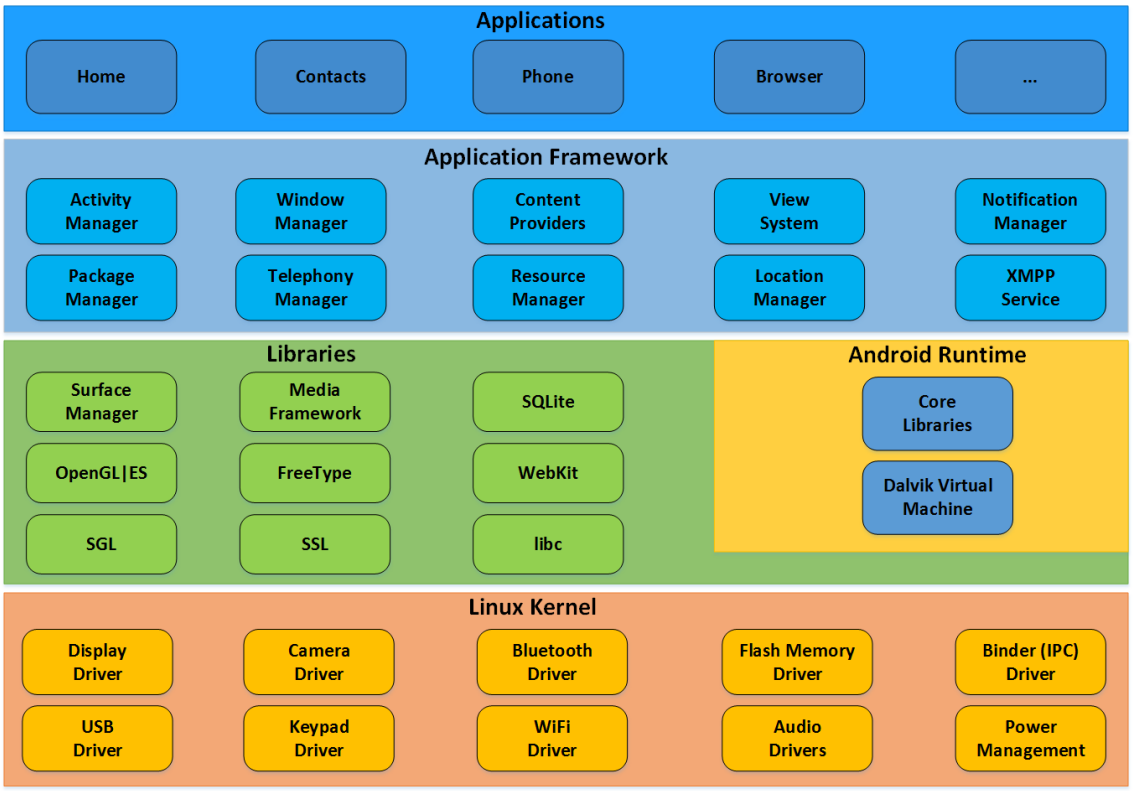
Osasyy tähän täysin ilmaiseksi saatavilla olevan käyttöjärjestelmän suosioon liittyy oletettavasti sen muokkausmahdollisuuksiin, sillä eri valmistajat pystyvät tuottamaan omia ohjelmistojaan sekä muokkaamaan järjestelmää vapaasti eri tuotantoon tarkoitettuja laiteperheitään varten [18]. Toinen merkittävä tekijä oli vuoden 2007 lopulla muodostettu Open Handset Alliance (OHA), jonka tarkoituksena oli kehittää avointa standardia mobiilijärjestelmille ja jonka jäsenistöön kuuluivat lähes kaikki suurimmat mobiilijärjestelmiä tuottavat yritykset, kuten esimerkiksi HTC, Qualcomm, Motorola ja NVIDIA [65, s. 2-4]. Vaikka Android-käyttöjärjestelmän lähdekoodi onkin pääasiassa Googlen tuottamaa, kaikki OHA:n alaiset yritykset osallistuvat säännöllisesti järjestelmän kehittämiseen [65, s. 2].

Järjestelmän huomattavasta suosiosta johtuen mobiilipelimaailma kiinnostui Androidista välittömästi sen julkaisun jälkeen. Erään tilaston mukaan jopa 65 prosenttia [48, s. 1] älypuhelinten 1.44 miljardista käyttäjästä [56] käyttää älypuhelintaan joskus pelaamiseen. Käyttöjärjestelmän perusohjelmistoihin kuuluva Google Play -sovelluskauppa (entinen Android Market) mahdollistaa hakemisen yli 1.4 miljoonan sovelluksen joukosta [55] ja vuoden 2014 aikana pelaajat käyttivät pelkästään mobiilipeleihin rahaa 11.07 miljardin dollarin edestä ympäri maailman [54]. Sekä älypuhelinten että pelaajien määrän ja peleistä saatavien tuottojen vuosittainen kasvaminen ei toistaiseksi näytä edes hidastuvan.

2.2 Android-järjestelmän arkkitehtuuri

Android-järjestelmän arkkitehtuuri on esitetty kuvassa 2.2. Kuvaa luettaessa tulee ymmärtää, että järjestelmä rakentuu kuvan mukaisesti alhaalta ylös ja korkeamman tason palvelut hyödyntävät edellisen kerroksen palveluita toimiakseen.

Android pohjautuu Linuxin ytimeen (kernel), joka sisältää ajurit käytössä olevalle laitteistolle, kuten esimerkiksi näytölle, näppäimistölle, kameralle, verkkokortille ja äänentoistolle. Lisäksi ydin on vastuussa muun muassa muistin- ja prosessinhallinnasta, verkkoliikenteestä, sekä tiedostojärjestelmän ylläpidosta. Ytimen päällä olevat natiivit järjestelmäkirjastot (system libraries) sisältävät toteutukset laskentakriittisille toiminnoille, joita ei kannata ajaa Dalvik-virtuaalikoneen kautta, tämän ollessa pääasiallinen ajonaikainen ympäristö sovelluskaupoista ladatuille sovelluksille. Nämä järjestelmäkirjastot tarjoavat perustoiminnot kolmannen kerroksen ohjelmistokehykselle (application framework) sekä Androidin ajonaikaisille toiminnoille. Androidin ohjelmistokehys tarjoaa lopulta Java-rajapinnan sovelluksille sekä kehittäjille järjestelmän sovelluskehitystä varten. [18]



Kuva 2.2: *Yleiskuva Android-järjestelmän arkkitehtuurista [31]*

Tavallisimmat Android-sovellukset suoritetaan toisen ja kolmannen kerroksen välissä sijaitsevalla Dalvik-virtuaalikoneella [65, s. 5-8]. Tämä virtuaalikone suorittaa tavukoodia, jollaiseksi Java-tiedostoista tuotetut .class-tiedostot käännetään arkkitehtuurikohtaisilla työkaluilla. Virtuaalikone on välittömässä vuorovaikutuksessa järjestelmäkirjastojen kanssa, jolloin tavukoodista tulkatut kutsut ohjataan tarvittaville natiivifunktioille toimintojen suorittamista varten. Androidin aiemmissa versioissa kaikki tavukoodi tulkattiin edellämainitulla tavalla, mutta nykyisin sen on syrjäyttämässä ART (Android Runtime) ajonaikainen ympäristö, jossa sovellus käännetään jo asennusvaiheessa kyseisen arkkitehtuurin konekielelle suorituskyvyn parantamiseksi sekä virran säästämiseksi [61].

2.3 Natiiviohjelmien kääntäminen ja suorittaminen Androidilla

Vaikka suurin osa Android-sovelluksista suoritetaankin Dalvik-virtuaalikoneella, käyttöjärjestelmällä voidaan ajaa myös natiivikoodilla (C/C++) toteutettuja osia sovelluksien sisällä [53, s. 27-54]. Tällöin ohitetaan arkkitehtuurin kolmannen kerroksen Java-ohjelmistokehys ja ollaan välittömässä yhteydessä toisen kerroksen järjestelmäkirjastojen kanssa. Natiivikoodilla toteutettuja ohjelmia ei ole tarkoitettu ajettavak-

si sellaisinaan yksittäisinä sovelluksina vaan se mahdollistaa suorituskykykriittisten osioiden toteuttamisen ja kääntämisen osaksi virtuaalikoneella suoritettavia Java-ohjelmia [65, s. 633-635].

Natiivikoodilla toteutettuja funktioita kutsutaan Javan natiivi-rajapinnan (JNI, Java Native Interface) kautta. Yhteys on kaksisuuntainen, eli myös Javalla toteutettava erikseen määriteltävä julkinen rajapinta on kutsuttavissa natiivikoodilla tuotetuilta osioilta. Natiivikielellä tuotetut kooditiedostot käännetään Android NDK (Native Development Kit) -työkalulla jaetuksi kirjastoksi (shared library), jotka ladataan virtuaalikoneen saataville ohjelman käynnistyessä. [65, s. 640-643]

2.4 Androidin erot työpöytäkäyttöjärjestelmistä

Android-käyttöjärjestelmän pohjautuessa Linux-ytimeen sen perustoiminnot työpöytäjärjestelmien kanssa ovat pitkälti yhteneviä. Suurimmat erot järjestelmien välillä muodostuvatkin muistinhallinnasta virransäästöominaisuuksien vuoksi. Tästä johtuen toisin kuin tavalliset Windows- tai Unix-prosessit, Android-sovellukset eivät täysin kontrolloi omaa elämänkaartaan (lifecycle) [40].

Jokaista Android-sovellusta ajetaan omassa erillisessä virtuaalikoneessaan. Tällaisia virtuaalikoneita voi olla useita rinnakkaisia aktiivisina jokaista käynnissä olevaa sovellusta varten. Kun järjestelmän resurssit ovat lopussa, näitä ajettavia sovelluksia joudutaan mielivaltaisesti sulkemaan. Tällaisissa tilanteissa järjestelmä ottaa huomioon seuraavat tekijät: [40]

- Käynnissä olevien ohjelmien määrä ja ajoaika
- Sovelluksen suhteellinen tärkeys käyttäjälle
- Vapaana olevan muistin määrä

Muistinhallinnasta johtuen kehittäjillä ei ole täydellistä kontrollia ohjelmien elinkaaren hallintaan. Käyttäjälle järjestelmän muistinhallinta ei näy muutoin, kuin että aiemmin käynnistetty sovellus voidaan ajoittain palauttaa muistista aiempaan tilaansa, kun taas joskus se joudutaan käynnistämään alusta asti uudelleen. Kehittäjien tulee vastaavasti reagoida ohjelman mielivaltaiseen sulkeutumiseen, esimerkiksi tallentamalla käynnissä olevan pelin tila käyttäjän poistuessa sovellusnäkymästä. [1]

Toinen mainitsemisen arvoinen eroavaisuus työpöytäjärjestelmistä on sovellusten tai niiden osien toteuttaminen joko aktiviteetteina (Activity) tai palveluina (Service).

Tavallisesti Android-sovellukset koostuvat aktiviteeteista, jotka vastaavat yksittäisiä käyttöliittymänäkymiä sovelluksissa. Nämä näkymät voivat kommunikoida keskenään ja käynnistää uusia aktiviteetteja jopa täysin eri sovelluksista, kuten esimerkiksi tapauksessa, jossa käyttäjä haluaa jakaa ottamansa valokuvan sähköpostiohjelman uusi viesti -toiminnon kautta. Palvelut puolestaan on tarkoitettu suorittamaan pitkäkestoisia tehtäviä taustalla, kuten esimerkiksi toistamaan musiikkia tai hakemaan dataa verkosta, eivätkä ne sisällä käyttöliittymänäkymää. Aktiviteetit voivat luoda uusia ja sulkea käynnissä olevia palveluita sekä vaihtaa dataa käynnissä olevien palveluiden kanssa. [1]

2.5 Mainokset ja IAP

Android mahdollistaa useiden eri liiketoimintamallien hyödyntämisen sovellusten monetisointiin, joista oleellisimmat on esitetty taulukossa 2.1 [48, s. 365-366]. Esitetyt mallit eivät ole toisiaan poissulkevia ja tarjottavasta sovelluksesta voidaan julkaista eri versioita eri käyttötarkoitukseen, kuten esimerkiksi ilmainen mutta mainosrahoitteinen versio tai vaihtoehtoisesti maksullinen ja mainokseton versio. Jotkut mallit vaativat myös muutoksia sovelluksen koodiin toimiakseen.

Taulukko 2.1: *Tarjolla olevat liiketoimintamallit [48, s. 366]*

Malli	**Tuotto**	**Vaatimukset**
Ilmainen	Saavuttaa helposti kattavan yleisön. Luo nimeä yritykselle.	-
Mainokset	Yritys saa rahaa jokaisesta mainosklikkauksesta.	Mainoskomponenttien sisällyttäminen sovellukseen.
IAP	Yritys saa rahaa pelaajan ostaessa virtuaalivaluuttaa tai muita pelissä tarvittavia hyödykkeitä.	Komponenttien sisällyttäminen sovellukseen, myytävien tuotteiden sekä hintojen miettiminen ja toteutus.
Maksullinen	Yritys saa rahaa pelaajan ostaessa ja ladatessa pelin.	-
Kuukausimaksullinen	Käyttäjät maksavat kiinteän kuukausimaksun ladatakseen sovelluskaupasta haluamiaan tuotteita	Tavallisesti sovelluskaupan ylläpitäjän vastuulla huolehtia käyttäjiltä saatavien tuottojen jakamisesta sovelluskehittäjille

Hyperkani Oy -yrityksen ansaintamalli perustuu pääasiassa käyttäjille pelien ohessa

näytettäviin mainoksiin sekä pelien sisäisiin ostoihin (In-App Purchases, IAP), joiden avulla pelaaja voi ostaa esimerkiksi kultaa, varusteita tai muuta pelissä käytettävää valuuttaa. Mainokset puolestaan koostuvat pelin ohessa näytettävistä bannereista sekä koko ruudun täyttävistä mainoksista, joissa molemmissa voidaan esittää joko yksittäisiä still-kuvia tai vaihtoehtoisesti lyhyitä videoita. Mainostettavat tuotteet vaihtelevat tavallisesti erilaisista peleistä esimerkiksi paikallisen pikaruokaketjun tarjouksiin, jälkimmäisen vaatiessa käyttäjältään lupaa eri tasoisten sijaintitietojen keräämiseen GPS:n (Global Positioning System) sekä Wlan-yhteyden avulla.

2.6 Android-laitteiden verkko-ominaisuudet

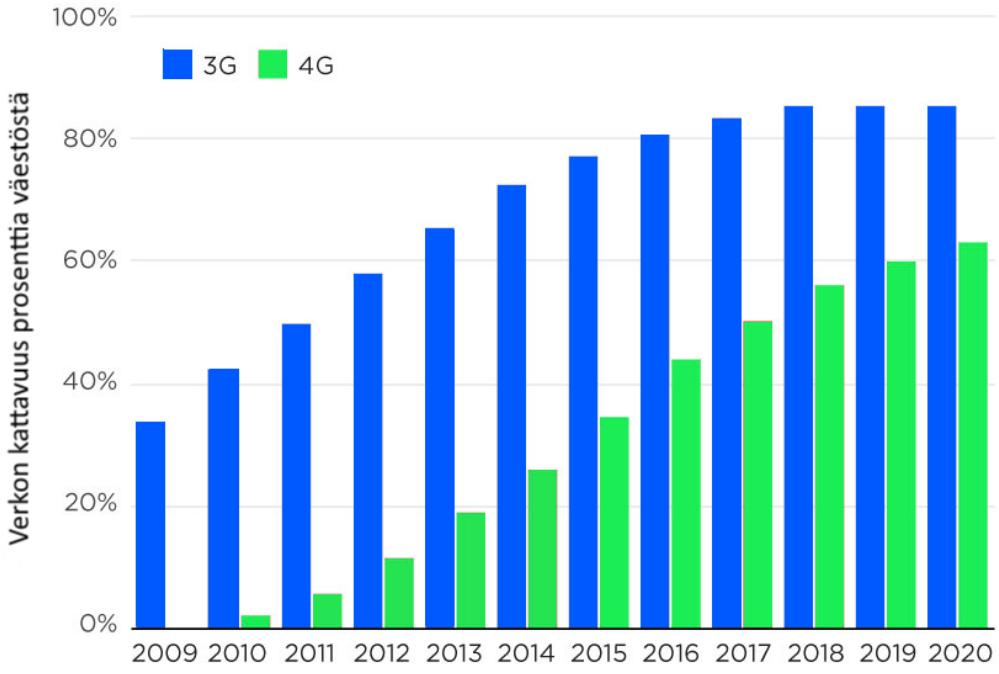
Vaikka esiasennetussa Android-järjestelmässä onkin tavallisesti mukana kaikki tarpeellinen käyttöjärjestelmän päivittäiseen käyttöön, kiinteällä internet-yhteydellä varmistetaan uusimpien päivitysten saatavuus reaaliaikaisesti sekä esimerkiksi tilitietojen jatkuva synkronointi ja varmuuskopiointi Googlen-palvelimille [65, s. 11-17]. Lisäksi useat sovellukset vaativat verkkoyhteyden toimiakseen. Tästä johtuen lähes kaikista laitteista löytyy tavallisen SIM-lukijan ja mobiiliradion lisäksi WLAN-komponentti langattomiin lähiverkkoihin liittymistä varten. [65, s. 11-17]

2.6.1 Eri verkkotekniikat pelikäytössä

Taulukossa 2.2 listataan tavallisimmat Android-laitteissa saatavilla olevat internetyhteyden mahdollistavat mobiililiittymien verkkotekniikat lähteen [29, s. 99-110] mukaisesti. Näistä reaaliaikaiseen pelikäyttöön soveltuvat havaintojen perusteella ainoastaan kolmannen ja neljännen sukupolven yhteystyypit, sillä 2G:n tapauksessa tiedonsiirtonopeudet ovat pelikäyttöön varsin rajoittuneita ja latenssit suuria. On huomattava, etteivät eri sukupolvien tiedonsiirtonopeudet tai latenssit suinkaan rajoitu taulukossa esitetyille arvoille, vaan nämä arvot antavat ainoastaan summittaisen mutta kuitenkin vertailukelpoisen kuvan jokaisen yhteystyypin suorituskyvystä.

Taulukko 2.2: *Verkkotekniikat Android-laitteissa [29, s. 100]*

Sukupolvi	**Yhteystyppejä**	**Nopeusluokka**	**Latenssi**
2G	GPRS, EDGE	100-400 Kb/s	300-1000 ms
3G	UMTS, HSPA, LTE	0.5-5 Mb/s	100-500 ms
4G	LTE-Advanced, HSPA+Rev11+	1-50 Mbit/s	$<$ 100 ms



Kuva 2.3: *3G- ja 4G-verkkojen levinneisyys [62]*

Kuvan 2.3 perusteella voidaan kuitenkin todeta sekä 3G- että 4G-verkkojen levinneisyys jo varsin kattavaksi [62] ja siksi näitä voidaankin pitää mobiilipelaajien keskuudessa oletusyhteystyyppeinä . Hyperkani Oy -yrityksen havaintojen perusteella molempien näiden verkkotyyppien latenssit pysyvät alhaisina ja sekä lataus- että lähetysnopeudet riittävinä nykyaikaista verkkopelaamista varten. Lisäksi eri analytiikkatyökaluilla saadun datan mukaan monet verkkopelaajat hyödyntävät myös WLAN-yhteyksiä verkkopelaamiseen.

2.6.2 NAT mobiiliverkoissa

Nopeat ja kehittyneet mobiiliverkot hyödyntävät laajasti osoitteenmuunnoksia (NAT) sekä palomuureja asiakaslaitteiden ja julkisen internetin välissä [63]. Tämän avulla estetään IPv4-osoitteiden loppuminen kesken sekä toisaalta suojellaan käyttäjiä, sillä ulkoapäin tulevat yhteyspyynnöt eivät tavallisesti pääse asiakaslaitteisiin asti vaan pysähtyvät palomuurien sekä osoitteenmuunnosten tarjoamaan suojaan. Tämä tarkoittaa kuitenkin myös sitä, että esimerkiksi suorien yhteyksien muodostaminen mobiiliverkkoihin yhdistyneiden laitteiden välille voi olla ongelmallista.

Osoitteenmuunnosten sekä palomuurien käyttäminen mobiiliverkoissa on täysin operaattorin vastuulla. Suomen verkoista esimerkiksi Elisa ei vuonna 2007 hyödyntänyt

osoitteenmuunnoksia ollenkaan, kun taas Soneralla oli käytössään sekä osoitteenmuunnos että tiukaksi asetettu palomuuri. Vastaavasti Yhdysvaltojen operaattorijätti AT&T:n asiakkaat sijaitsivat osoitteenmuunnosten takana ja toisaalta Taiwanista löytyi operaattori, jolla ei ollut palomuuriakaan käytössä. [42]

Suorien yhteyksien muodostamiseen osoitteenmuunnosten takana olevien tahojen välille on kuitenkin kehitelty erilaisia menetelmiä, joiden onnistuminen riippuu sekä osoitteenmuunnoksen että palomuurin asetuksista. Näitä menetelmiä käsitellään myöhemmin kohdassa 8.1.

2.7 Moninpelit mobiililaitteilla

Sekä reaaliaikaiset että vuoropohjaiset verkon yli muiden pelaajien kanssa pelattavat moninpelit, kuten esimerkiksi Clash of Clans, Words with Friends ja Minecraft, ovat yleistyneet mobiilipelimaailmassa merkkittävästi. Osatekijöinä tähän ilmiöön on vaikuttanut sekä kolmannen ja neljännen sukupolven mobiiliverkkojen laaja levinneisyys että operaattoreiden jatkuva kilpailu dataliikenteen kustannuksista. Pääasiassa pelitarjontaan vaikuttavat kuitenkin suurimmilta osin kuluttajien eli pelaajien tarpeet. PC- sekä konsolimarkkinoilla moninpeliominaisuudet ovat lukeutuneet pelien perustoiminnallisuuksiin jo vuosikymmen sitten, mutta mobiilimaailmassa tämä kaikki on jälleen uusinta teknologiaa, johon mobiilipelaajat ympäri maailman haluavat päästä käsiksi. [48, s. 1-5][65, s. 18-20]

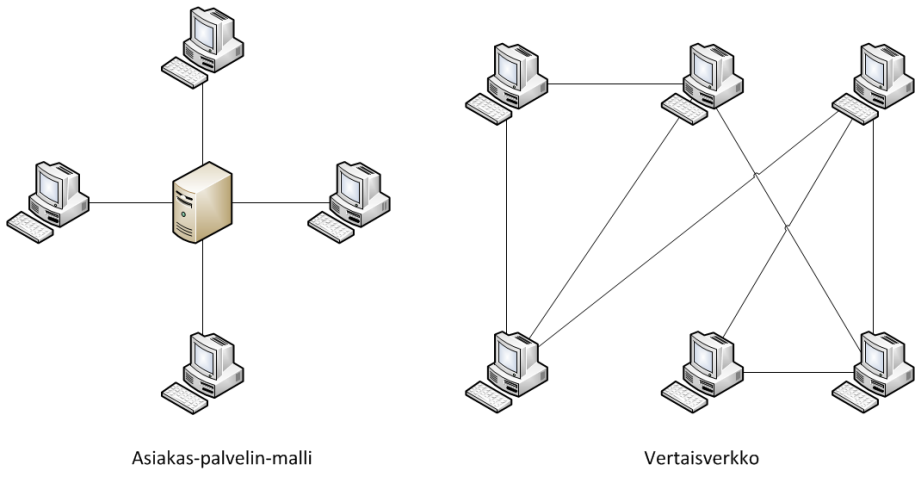
Mobiilimoninpelien yleistymistä vauhdittaa myös esimerkiksi Googlen julkaisema Google Play Games Services Realtime Multiplayer API, joka vastaa reaaliaikaisten moninpelien vaatimista verkkoyhteyksistä sekä peliseuran etsimisestä muiden pelaajien keskuudesta [24]. Vastaavanlainen järjestelmä sisältyy myös laajasti käytettyyn Unity-pelinkehitysympäristöön [60]. Madaltamalla mobiilimoninpelien kehittämisen vaatimaa oppimiskynnystä sekä abstrahoimalla moninpelien vaatima tekninen toiminnallisuus erillisten valmiiden ohjelmistokehysten taakse, pelinkehittäjien on varsin helppo huomioida jatkossa myös mobiilimoninpelaajien tarpeet projekteissaan.

3. VERTAISVERKKOJEN TOIMINTAPERIAATTEET

Tämä luku keskittyy vertaisverkkojen toimintaperiaatteisiin sekä niiden suurimpiin hyötyihin ja tyypillisimpiin ongelmakohtiin verkkopelaamisen näkökulmasta.

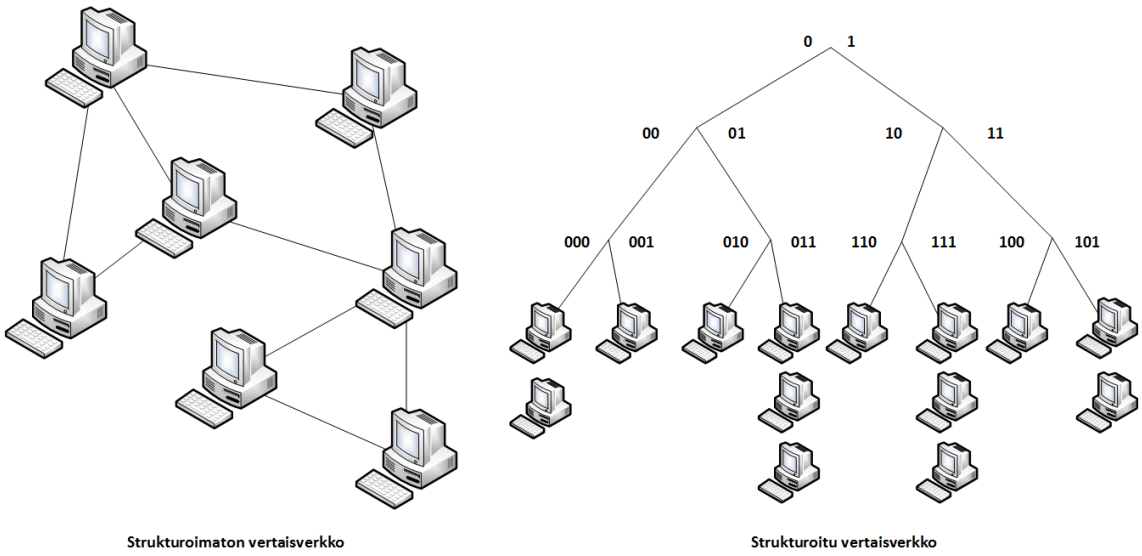
3.1 Vertaisverkkojen arkkitehtuuri

Vertaisverkon (Peer-to-Peer, P2P) määritelmällä tarkoitetaan sellaista verkkotopologiaa, jossa verkon jokainen taho tarjoaa omia resurssejaan, kuten esimerkiksi laskentatehoa tai levytilaa, käytettäväksi muille verkon jäsenille ilman erillistä keskuspalvelinta, ja toimii siten sekä asiakkaana että palvelimena (Servent) osana verkkokokonaisuutta [52]. Määritelmän mukaan vertaisverkko eroaa tässä suhteessa toistaiseksi yleisimmästä internetissä käytetystä arkkitehtuurimallista, asiakas-palvelinmallista, jossa verkkoon sijoitetut laitteet toimivat joko asiakkaina tai palvelimina mutta eivät yhtäaikaisesti molempina. Vertaisverkko mahdollistaa resurssien hajauttamisen sekä osaltaan vastaa tulevaisuuden internetin vaatimuksiin: skaalautuvuuteen, tietoturvaan, luotettavuuteen, joustavuuteen sekä palvelun laatuun [58]. Kuva 3.1 havainnollistaa erään vertaisverkon sekä asiakas-palvelin-mallin oleellisinta arkkitehtuurillista eroavaisuutta.



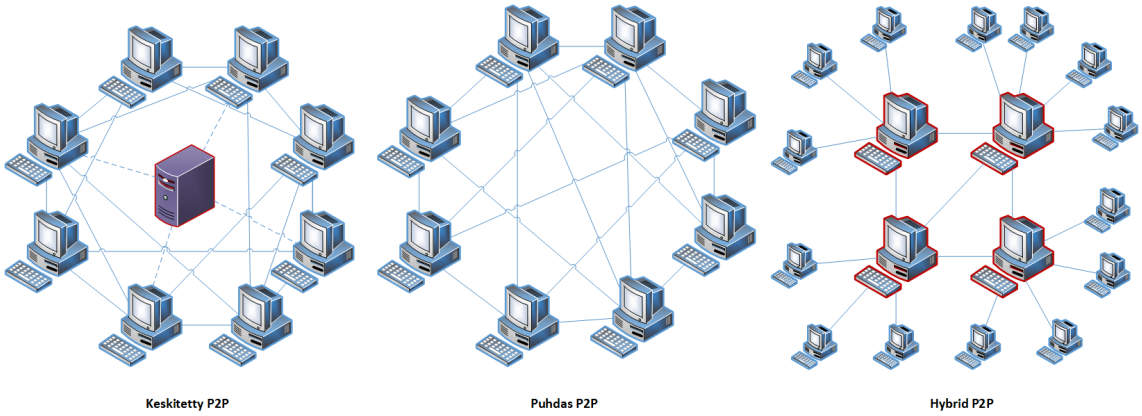
Kuva 3.1: *Asiakas-palvelin-malli sekä vertaisverkko*

Vertaisverkot mahdollistavat sisällön lähettämisen suoraan sitä tarvitsevalle taholle ilman, että sitä tarvitsee kierrättää erillisen keskuspalvelimen kautta [52]. Verkkopelaamisen näkökulmasta tämä tarkoittaa pienempiä vasteaikoja ja siten reaaliaikaisempaa pelikokemusta, eikä järjestelmä ole tällöin riippuvainen kiinteästä keskuspalvelimesta, johon kohdistuvat häiriötilanteet tekevät palvelusta käyttökelvottoman [43]. Tämä tulee toisaalta huomioida sovelluksen kehityksessä järjestelmän eheyden sekä tietoturvan näkökulmasta, sillä toisin kuin asiakas-palvelin-mallin arkkitehtuurissa, tahojen välissä ei ole kiinteää keskuspalvelinta, joka vastaisi saapuvan sisällön laillisuudesta tai oikeellisuudesta [33][41].



Kuva 3.2: *Strukturoimaton sekä strukturoitu vertaisverkko*

Vertaisverkot jaotellaan strukturoituihin ja strukturoimattomiin vertaisverkkoihin. Strukturoiduissa vertaisverkoissa tahojen väliset yhteydet muodostuvat erillisten sääntöjen mukaan noudattaen esimerkiksi puumaista topologiaa, kun taas strukturoimattomissa vertaisverkoissa yhteydet muodostuvat täysin epäsäännöllisesti. Strukturoimattomat vertaisverkot kestävät huomattavasti strukturoimattomia verkkoja paremmin suurten käyttäjämäärien jatkuvia verkkoon liittymisiä ja verkosta lähtemisiä, mutta tuottavat toisaalta ongelmia erityisesti harvinaisempia resursseja etsittäessä, jolloin resurssikysely tulee lähettää koko vertaisverkon yli. Strukturoidut vertaisverkot ratkaisevat tämän ongelman tavallisesti käyttämällä hajautettuja hajautustauluja (distributed hash table), mutta kestävät huonommin verkon rakenteeseen kohdistuvia muutoksia. Kuvassa 3.2 esitetään strukturoimaton sekä strukturoitu vertaisverkko. [38]



Kuva 3.3: *Eri vertaisverkkotyypit*

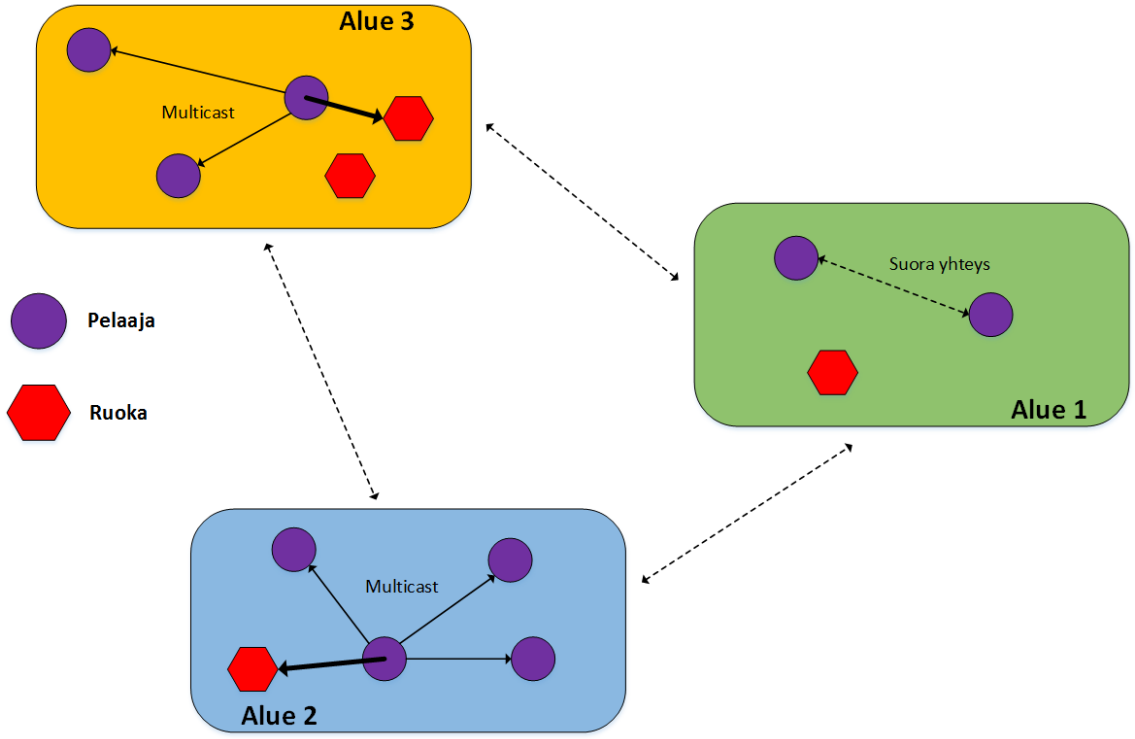
Strukturoimattomat vertaisverkot voidaan lisäksi eritellä sekä puhtaisiin- että hybridivertaisverkkoihin [52]. Jälkeenpäin tulkintaan on lisätty myös keskitetyt vertaisverkot [58]. Tulkinnan mukaan keskitetyssä vertaisverkossa käytetään erillistä havaintopalvelinta ylläpitämään verkon eri tahojen osoitteita sekä jaettuja resursseja, kun taas puhtaassa vertaisverkossa tahot muodostavat yhteyksiä sekä etsivät resursseja keskenään mielivaltaisesti. Hybridivertaisverkko on kompromissi näiden kahden arkkitehtuurin väliltä, jossa verkon keskussolmut valitaan dynaamisesti verkon kulloinkin saatavilla olevista tahoista. Kuva 3.3 havainnollistaa näiden vertaisverkkotyyppien eroavaisuuksia.

Lisäksi vertaisverkkojen toimintaan kuuluu olennaisena osana peiteverkon (Overlay network) käyttäminen, jossa määritellään olemassa olevat kommunikointiyhteydet eri tahojen välillä sekä eri tahoilta saatavissa olevat resurssit [4]. Peiteverkot jaotellaan deterministisiin sekä epädeterministisiin peiteverkkoihin niiden toimintaperiaatteiden perusteella, joista ensimmäisiä käytetään strukturoitujen vertaisverkkojen ja jälkimmäisiä strukturoimattomien vertaisverkkojen yhteydessä [58].

Verkkopelaamisen näkökulmasta vertaisverkkojen käyttöön liittyy hyvin usein ainakin yksi erillinen kehittäjien ylläpitämä havaintopalvelin, joka pitää kirjaa jo verkkoon kuuluvien käyttäjien osoite- sekä mahdollisista pelienhakutiedoista ja jakaa ne tarvittaessa eteenpäin uusille verkon käyttäjille [3]. Tällaisia palvelimia kutsutaan matchmaking-palvelimiksi. Vertaisverkkotuen lisäksi ne saattavat toimia rajapintana pelaajien profiilitietoihin ja kaverisuhteiden ylläpitoon sekä tarjota esimerkiksi julkisia keskusteluhuoneita muiden pelaajien välillä, mutta niiden pääasiallinen tehtävä on tehdä pelien löytämisestä mahdollisimman helppoa pelaajien kannalta kätkemällä hakuprosessiin vaikuttavia tekijöitä [6].

3.2 Vertaisverkkojen tarjoamat hyödyt verkkopelaamisen näkökulmasta

Eräs suurimmista vertaisverkkojen tarjoamista hyödyistä verkkopelaamisen näkökulmasta on asiakas-palvelin-malliin verrattuna parantunut suorituskyky, joka näkyy pelaajille pienempinä viiveinä [41][43]. Viiveitä mitataan tavallisimmin millisekunneissa ja viiveen suuruus kertoo, kauanko toisen pelaajan toiminnoilla kestää synkronoitua muille pelaajille [44]. Viiveen vaikutus pelikokemukseen on tietenkin riippuvainen pelityypistä, sillä vaikka esimerkiksi jotkut strategiapelit kestävätkin jopa muutamien sekuntien viiveitä, saattaa reaaliaikaisen ensimmäisessä persoonassa kuvatun toimintapelin pelaaminen kärsiä jo pienilläkin viiveillä [2][10]. Huomionarvoista on myös se, että myös muiden pelaajien pelikokemus kärsii yksittäisen pelaajan viiveiden kasvaessa [32].



Kuva 3.4: *Vertaisverkon osittaminen pelialueen perusteella [34]*

Suorituskyky on kuitenkin riippuvainen pelaajien eli verkon tahojen määrästä. Pienet, tavallisimmin alle 40 pelaajan väliset vertaisverkot mahdollistavat kaiken tiedon välittämisen kaikkien tahojen välillä ilman suoranaista vaikutusta suorituskykyyn [33], mutta verkkojen kasvaessa tulee joko lähetettävän tiedon määrää tai vastaanottajien lukumäärää supistaa [11]. Tavallisimmin tämä toteutetaan jakamalla

verkkopeli esimerkiksi pelialueen perusteella osiin, joissa pelaajien tuottamat toiminnot välitetään ainoastaan niiden välittömässä läheisyydessä oleville pelaajille [11][16][41]. Kuva 3.4 havainnollistaa verkon jakamista pelialueen mukaisesti.

Edellämainittu pelialueellinen jakaminen tuo kuitenkin esiin toisen vertaisverkkojen tarjoaman hyödyn, skaalautuvuuden. Tällaiseen verkkopeliin voi parhaimmillaan liittyä tuhansia yhtäaikaisia pelaajia ilman, että kenenkään pelikokemus kärsii suuresta pelaajamäärästä aiheutuvien viiveiden vuoksi [16][34]. Mikäli jokainen näistä pelaajista ylläpitäisi jatkuvaa yhteyttä kiinteään pelipalvelimeen asiakas-palvelinmallin mukaisesti ja välittäisi tälle kaiken vertaisverkossa ylläpidettävän tiedon, vaatimukset palvelinten suorituskyvylle, yhteyksien kapasiteetille sekä pelin kompleksisuudelle luonnollisesti kasvaisivat [41][43].

3.3 Vertaisverkkoihin liittyvät ongelmat

Vaikka vertaisverkolla saavutetaankin selkeää suorituskykyetua pelikäytössä asiakas-palvelin-malliin verrattuna, liittyy vertaisverkkojen käyttöön kuitenkin joitakin ongelmia [33]. Jotkut ongelmista esiintyvät vertaisverkkojen yhteydessä ainoastaan peleissä, mutta suurin osa ongelmista on yleistettävissä myös muihin vertaisverkkojen käyttökohteisiin. Yleisimmät ongelmat liittyvät tavallisesti yhteyden muodostamiseen, tietoturvaan, huijaamiseen, pelin tilan ylläpitoon sekä viiveiden hallintaan. [43]

Ensimmäinen vertaisverkon käyttöönottoon liittyvä ongelma voi esiintyä jo yhteyksiä luotaessa: mikäli verkkoon liityvä taho on kytkimen, reitittimen, palomuurin tai vastaavan osoitteenmuunnoksen (NAT, Network Addres Translation) toteuttavan laitteen takana, ei kyseiseen tahoon pysty sellaisenaan muodostamaan yhteyttä ulkoverkosta [12]. Osoitteenmuunnosta hyödynnetään piilottamaan aliverkossa sijaitsevia laitteita, jolloin kyseiset verkkoon kytketyt laitteet eivät omaa julkista IP-osoitetta [14]. Mikäli verkon muut osapuolet ovat julkisesti näkyvillä, NAT:n takana oleva laite pystyy kuitenkin muodostamaan näihin yhteyden [47]. Useamman käyttäjän välisessä verkossa osoitteenmuunnoksen takana olevat tahot eivät voi keskenään muodostaa yhteyksiä, mutta sisältö voidaan kierrättää julkisten verkon tahojen kautta, joskin tällöin osa vertaisverkon mahdollistamasta suorituskyvystä menetetään [12]. Ongelman ratkaisuksi on määritelty erillinen ICE-protokolla, joka sisältää erilaisia menetelmiä yhteyden muodostamiseksi ja sen toimintaperiaatetta käsitellään myöhemmin tässä luvussa.

Toinen vertaisvertaisverkkoihin liittyvä ongelma muodostuu vertaisverkkojen puutteellisesta tietoturvasta ja tämä altistaa verkkopelaajat huijaamiselle [43]. Huijaa-

miseksi voidaan luokitella sellainen toiminta, jossa käyttäjä rikkoo pelin sääntöjä ja saavuttaa toiminnallaan hyötyä kanssapelaajiinsa nähden. Koska vertaisverkossa pelattavasta pelistä puuttuu kiinteä keskuspalvelin, sisällön oikeellisuuden ja järkevyyden tarkastaminen on muiden verkon tahojen vastuulla. Tästä johtuen pelin kehittäjien tuleekin toteuttaa pelin logiikkaan vaaditut tarkastelut saapuvan datan oikeellisuuden tarkastamiseksi. [16][34]

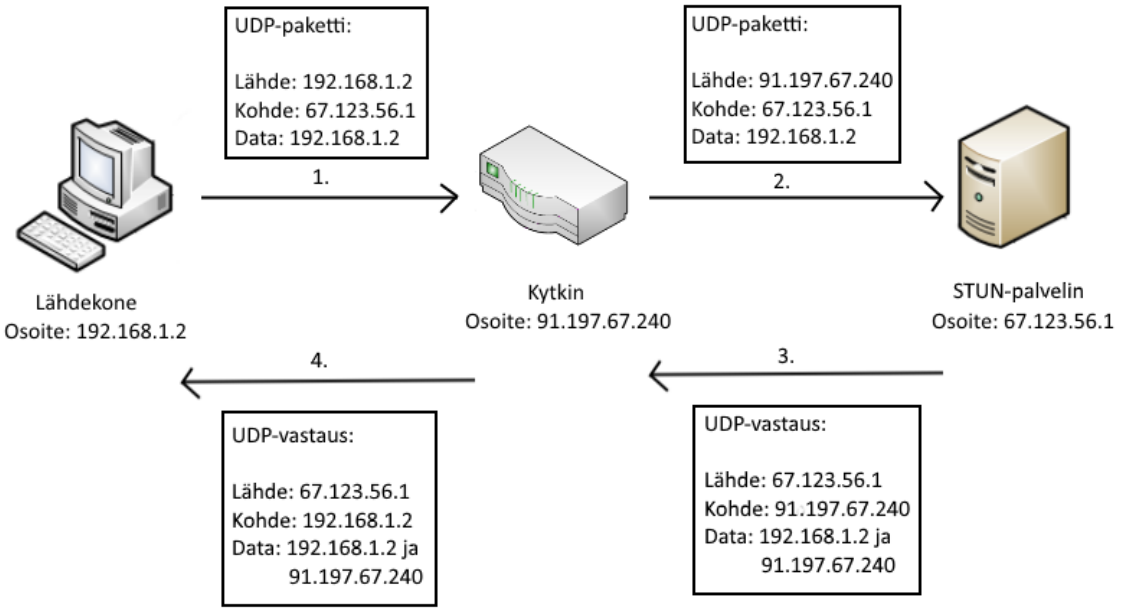
Kehittäjien vastuulle jää myös pelin tilan ylläpito sekä sen reaaliaikainen synkronointi. Viiveistä, kadonneista paketeista sekä yhteyksien mahdollisista katkeamisista johtuen on mahdollista, että verkon eri tahoilla peli on ajoittain eri tiloissa [44]. Ongelmia muodostuu esimerkiksi tilanteissa, jossa saapuvan paketin sisältämä tieto on riippuvainen sellaisesta tilatiedosta, mitä ei ole toistaiseksi vastaanotettu. Vertaisverkkoa hyödyntävän ohjelman tulee varautua tähän ongelmaan sekä toimia järkevästi kaikissa tilanteissa realistisen pelikokemuksen takaamiseksi. Tilatietojen synkronoinnin kestolla on siten selkeä vaikutus tämän ongelman esiintymiseen. Lisäksi vertaisverkon skaalautuvuudesta huolimatta suuret käyttäjämäärät samassa verkossa voivat aiheuttaa korkeampia viiveitä ja siten olla osaltaan syyllisiä tilatietojen synkronoinnin hitauteen [43]. [16]

3.4 Yhteyden muodostus & ICE-protokolla

Vertaisverkon pystytys aloitetaan yhteyksien luomisella haluttujen verkon tahojen välille. Näiden yhteyksien luomisen epäonnistuminen on kuitenkin eräs tavallisimmista vertaisverkkoihin liittyvistä ongelmista [9]. Tästä johtuen sen ratkaisuksi on kehitelty useita erilaisia menetelmiä sekä vaihtoehtoisia toteutustapoja. ICE-protokollalla (Interactive Connectivity Establishment) tarkoitetaan tällaisten tekniikoiden kokoelmaa sekä toteutusta ja näiden tekniikoiden avulla pyritään muodostamaan vertaisverkko haluttujen tahojen välille [29, s. 41-42][49].

Yhteyttä muodostettaessa on oleellista, että eri tahot ovat tietoisia siitä, ovatko ne osoitteenmuunnoksen tai muun vastaavan tekniikan takana. Tähän käytetään tavallisimmin STUN-protokollaa (Simple Traversal of UDP through NATs), jota noudattaessaan vertaisverkkoon liittyvä taho lähettää erilliselle julkisesti saatavilla olevalle STUN-palvelimelle oman IP-osoitteensa UDP-paketin sisältönä. Sama osoite liitetään UDP-paketin määritelmän [46] mukaisesti myös paketin otsikkotietoihin. Mikäli paketti kulkee osoitteenmuunnoksen kautta, otsikkotiedoissa oleva lähtöosoite vaihdetaan kyseistä laitetta vastaavaksi. Paketin saapuessa STUN-palvelimelle näitä kahta osoitetta verrataan keskenään ja mikäli niissä ei havaita eroavaisuuksia, voidaan varmasti todeta tämän tahon olevan julkisesti tavoitettavissa. STUN-palvelin

vastaa tähän pakettiin sisällyttämällä myös otsikosta löytyneen osoitteen paketin sisällöksi alkuperäisen osoitteen tavoin. Kuva 3.5 havainnollistaa STUN-protokollan toimintaa. [49][29, s. 42]



Kuva 3.5: *Välissä olevan osoitteenmuunnoksen osoitteen selvittäminen STUN-protokollalla*

Toisaalta eräät osoitteenmuunnosmenetelmät on toteutettu siten, että auki olevien UDP-yhteyksien tilaa ei tallenneta lähdelaitteen aliverkotetun osoitteen sekä portin yhteydessä osoitteenmuunnoslaitteen reititystauluun. Tämä tarkoittaa sitä, että jatkossa kaikki tähän kyseiseen osoitteenmuuntajaan saapuvat UDP-paketit ohjataan samaiseen aliverkotettuun lähdelaitteeseen niiden lähettäjästä riippumatta. Tällöin riittää, että aliverkon takana oleva taho lähettää yhden STUN-paketin julkiselle STUN-palvelimelle portin avaamiseksi sekä käytössä olevan osoitteenmuuntajan osoitteen selvittämiseksi. Vertaisverkon muut tahot voivat tämän jälkeen muodostaa UDP-yhteyden tähän osoitteenmuuntajaan, joka ohjaa kaikki saapuneet paketit alkuperäiselle lähdelaitteelle. Menetelmää kutsutaan nimellä *UDP hole punching* ja sitä käsitellään tarkemmin kohdassa 8.1.2. [12]

Mikäli edellämainittu osoitteenmuunnoksen suorittava laite ei kuitenkaan ohjaa eri osoitteista saapuvia UDP-paketteja lähdelaitteelle aiemman esimerkin mukaisesti, ei suoran vertaisverkkoyhteyden muodostaminen tähän laitteeseen tällöin onnistu ilman erillistä logiikkaa osoitteenmuunnoksen toimintaperiaatteen selvittämiseksi

[17]. Ratkaisuksi jää TURN-protokolla (Traversal Using Relay NAT), jossa erillinen julkinen TURN-palvelin toimii välittäjänä kahden osoitteenmuunnoksen takana olevan laitteen välillä. TURN-palvelin siis välittää ensimmäiseltä taholta saapuneet paketit toiselle taholle ja päinvastoin. Tämä takaa, että yhteys tahojen välille saadaan varmasti muodostettua, mutta koska paketit kiertävät erillisen julkisen palvelimen kautta, suoran yhteyden tuomat edut menetetään. Laajan käyttäjämäärän sisältämissä sovelluksissa TURN-palvelimiksi voidaan myös valjastaa sellaisia verkon tahoja, jotka ovat julkisesti saatavilla (esimerkiksi Skypen Super-peer) [12]. Tällöin vältytään erillisen palvelimen ylläpidolta sillä kustannuksella, että nämä asiakkaat joutuvat käsittelemään tavallista suurempia tietomääriä. [29, s. 42-44][49]

3.5 Vertaisverkkojen tavallisimpia käyttökohteita

Verkkopelaamisen ohella tavallisimpiin vertaisverkkojen käyttökohteisiin kuuluvat muiden muassa hajautettu laskenta, VoIP (Voice over IP) puhelinjärjestelmät, videoiden sekä musiikin suoratoisto, sisällön ja tiedostojen jakoon keskittyneet torrentverkot sekä esimerkiksi bitcoinit [52]. Monet näistä järjestelmistä perustuvat pääasiassa asiakas-palvelin-mallin arkkitehtuuriin käyttäen kuitenkin vertaisverkkoja esimerkiksi reaaliaikakriittisen sisällön jakamiseen usealle käyttäjälle samanaikaisesti tai hyödyntämään suuren vertaisverkon mahdollistamaa laskentatehoa.

4. KÄYTÖSSÄ OLEVA JÄRJESTELMÄ

Tällä hetkellä Hyperkani Oy toteuttaa reaaliaikaiset moninpelit Google Play Games Services -SDK:n [22] avulla. Google Play Games Services SDK sisältää moninpelirajapinnan lisäksi toteutukset julkisille pisteytystaulukoille (leaderboards), saavutuksille (achievements), tapahtumille ja tehtäville (events & quests) sekä datan tallentamiselle pilveen. Kyseinen SDK on saatavilla Android-järjestelmän ohessa myös iOS-järjestelmälle [25] ja lisäksi tarjolla on REST-rajapinta (Representational State Transfer) selainpohjaisia sovelluksia varten [23].

4.1 Moninpelirajapinnan toimintaperiaate

Google Play Games Services SDK:n moninpelirajapinnan toimintaperiaate perustuu vertaisverkkoihin, jolloin samassa pelissä olevat pelaajat viestivät keskenään ilman erillistä yhteyttä Google Play Games Services -palvelimille. Rajapintaa asiakasohjelman kautta käyttävä pelaaja yhdistetään aluksi Google Play Games Services -palvelimelle tunnistautumista varten. Tunnistaumisen jälkeen pelaaja voi vaihtoehtoisesti joko pyytää omia Google+-kavereitaan peliseuraksi, hyödyntää palvelimen matchmaking-järjestelmää satunnaisten pelikavereiden etsimiseksi tai yhdistellä näitä kahta mielivaltaisesti. Pelaajien löydyttyä kyseiset pelaajat yrittävät muodostaa *huoneen* eli vertaisverkon keskenään. Google Play Games Services SDK mahdollistaa enimmillään kahdeksasta pelaajasta koostuvien huoneiden luomisen. Lisäksi matchmaking-järjestelmälle voidaan asettaa rajoitteita muiden pelaajien etsimiseen liittyen, jolloin saadaan erotettua esimerkiksi sovelluksen eri versioiden käyttäjät keskenään. [24]

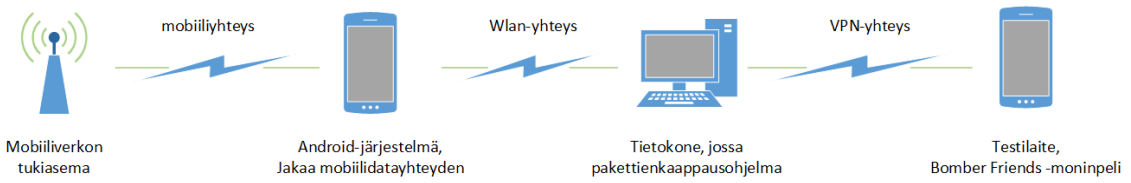
Huonetta luotaessa pelaajien välille pystytetään vertaisverkko sekä TCP-, että UDP-yhteyksillä, joista SDK:n dokumentaatiossa käytetään termejä *reliable* ja *unreliable*. Huone on onnistuneesti luotu ainoastaan silloin, kun verkon jokaisen tahon välille on saatu onnistuneesti muodostettua molemmat yhteydet, joko suoraan tai vaihtoehtoisesti erillisten relay-palvelinten välityksellä. Näistä ainoastaan TCP-yhteyden yli lähetetylle datalle taataan tiedonsiirron luotettavuus sekä pakettien saapuminen lähetysjärjestyksessä protokollan tilallisesta luonteesta johtuen [29, s. 13-32][45], kun taas UDP-yhteyksien yli lähetetyt paketit ovat teoreettisesti katsoen nopeampia,

mutta niiden perille saapumista ei taata ja niiden saapumisjärjestys voi vaihdella tahojen välillä [29, s. 35-38][46]. Verkkopeleissä TCP-yhteyttä hyödynnetään tavallisesti pelitilakriittisen tiedon siirtämiseen, johon luetaan esimerkiksi erilaiset pelimaailmaa muokkaavat tapahtumat, kun taas UDP-yhteydellä välitetään viivekriittisemmät osiot, kuten esimerkiksi pelaajien liikkuminen tai ampuminen, jolloin yksittäisten UDP-pakettien hukkuminen tai viivästyminen ei juurikaan vaikuta käyttäjien havaitsemaan pelikokemukseen [29, s. 30-32]. [24]

Google Play Games Services SDK:n moninpelirajapinnan ytimenä toimii avoimen lähdekoodin libjingle-kirjasto [19], joka toteuttaa ICE-määritelmän mukaiset protokollat vertaisverkkoyhteyksien luomiseen, neuvottelee käytettävät codecit ja formaatit esimerkiksi videoiden suoratoistoa verten sekä toimii datan välittäjänä. Muiden vertaisverkon tahojen etsimiseen libjingle hyödyntää XMPP-protokollaa [20], jonka alkuperäinen käyttötarkoitus on ollut pikaviestinnässä [51]. Libjingle-kirjaston muita käyttökohteita ovat esimerkiksi useita yhtäaikaisia käyttäjiä tukevat puhe- ja videoviestimet sekä reaaliaikaiset musiikintoisto- ja tiedostonjako-ohjelmat.

4.2 Toimintaperiaatteen tarkastelu verkon analysointityökalun avulla

Google Play Games Services -moninpelirajapinnan yksityiskohtaisemman toimintaperiaatteen tutkimiseksi pystytettiin testausympäristö, jossa testin kohteena oleva Android-laite yhdistettiin VPN-yhteydellä (Virtual Private Network, virtuaalinen erillisverkko) tietokoneelle, joka oli yhteydessä internettiin toisen Android-laitteen jakaman mobiiliverkkoyhteyden kautta kuvan 4.1 mukaisesti. Tällöin kaikki kohdelaitteen lähettämät ja vastaanottamat paketit kiertävät tämän tarkoitukseen soveltuvan tietokoneen kautta, jolloin näitä paketteja on mahdollista tallentaa ja tarkastella erillisen verkon analysointityökalun [64] avulla. Lisäksi muodostamalla verkkoyhteys toisen Android-laitteen jakaman mobiiliverkon kautta varmistetaan moninpelikomponentin toiminta tavallisella mobiilidatayhteydellä.



Kuva 4.1: *Testausympäristö moninpelirajapinnan yksityiskohtaisemman toiminnan selvittämiseksi*

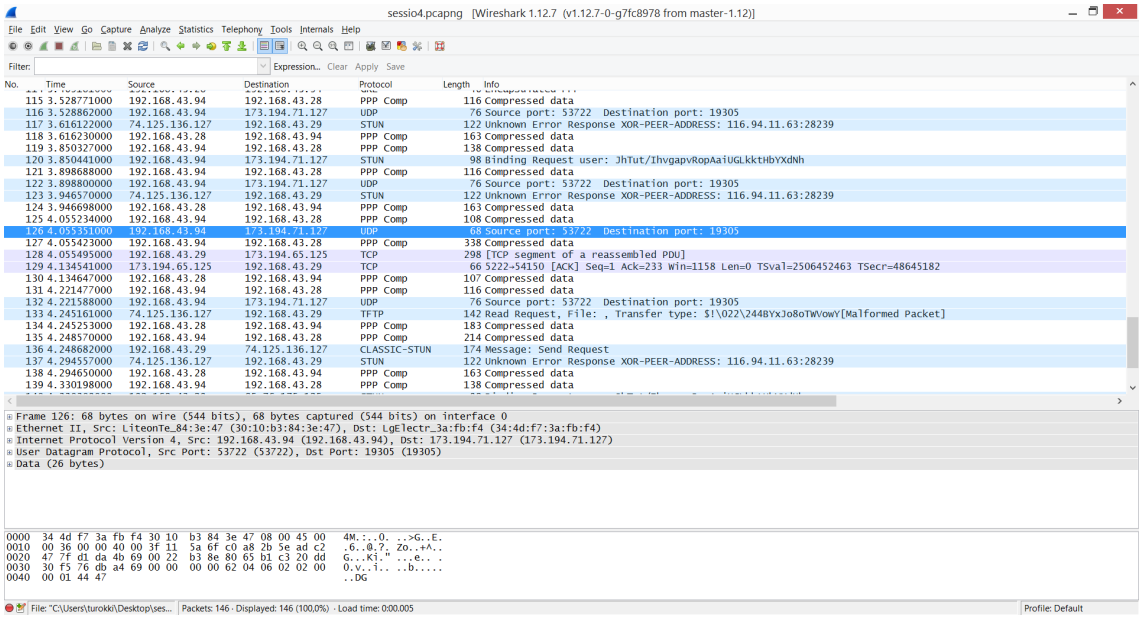
Testauskäytössä olleelle tietokoneelle asennettiin Wireshark-analysointiohjelma [64] kohdelaitteen verkkoliikenteen tarkastelua varten. Tämän jälkeen kohdelaitteella käynnistettiin Hyperkani Oy -yrityksen toteuttama Bomber Friends -moninpeli [27], joka hyödyntää Google Play Games SDK:ta moninpeliominaisuuksien toteuttamiseksi.

Nopeatempoisessa Bomber Friends -pelissä 2-4 pelaajaa yrittävät pommittaa toisiaan muuttuvassa pelimaailmassa, kunnes jäljellä on enää yksi elossa oleva pelaaja. Voittajan selvittyä aloitetaan uusi peli samojen pelaajien kesken. Peli on suhteellisen viivekriittinen, sillä tuhottavan maailman tila on hajautettu kaikkien pelaajien kesken ja viiveiden kasvaessa asiakasohjelmien pelitilojen välinen synkronointi voi epäonnistua. Tämä näkyy pelaajille siten, että osa pelaajista saattaa kulkea *seinien läpi* tilanteissa, joissa kyseisen seinän hajoaminen ei ole vielä synkronoitunut jollekin pelaajalle. Viiveiden minimoinnin vuoksi pelaajien väliset liikkeet ja sijainnit välitetään muille pelaajille UDP-viesteinä, kun taas maailmaa muokkaavat tapahtumat varmistetaan TCP-yhteydellä.

Wireshark-analysointityökalulla tallennettujen ja tutkittujen pakettien lähde- ja kohdeosoitteiden perusteella havaitaan, että Google Play Games SDK:n matchmaking-järjestelmä ei mitä todennäköisimmin käytä pelinhakijoiden maantieteellistä sijaintia hakukriteerinä pelaajia yhdistellessään. Vaikka Google Analytics-analytiikkatyökalun perusteella testausajanhetkellä yhtäaikaisia pelaajia oli useita tuhansia, pelit muodostettiin lähes poikkeuksetta siten, että huoneen jokainen pelaaja saattoi sijaita eri mantereella. On mahdollista, että järjestelmä etsii pelaajan ja matchmaking-palvelimen välisiä osoitteenmuunnoksia sekä palomuureja ja pyrkii sitten muodostamaan pelit siten, että ainakin osa pelaajista olisi julkisen internetin saavutettavissa vertaisverkon muodostamista varten.

Toinen mielenkiintoinen pakettien lähde- ja kohdeosoitteiden perusteella tehty havainto koskee vertaisverkkoyhteyden muodostamista. Lähes poikkeuksetta UDP-paketit saatiin onnistuneesti välitettyä pelaajien kesken vertaisverkkojen toimintaperiaatteiden mukaisesti, mutta kaikki TCP-paketit kiersivät pelaajista riippumatta samojen IP-osoitteiden kautta. Traceroute-työkalulla selvitettiin, että palvelimet sijaitsevat Kaliforniassa, Yhdysvalloissa. Keskenään samassa lähiverkossa sijaitsevat pelaajat onnistuivat kuitenkin välittämään myös TCP-paketit suoraan toisilleen ilman erillistä palvelinta. Tästä voidaan päätellä, ettei pelaajien välisen vertaisverkkoyhteyden muodostaminen osoitteenmuunnoksista ja palomuureista johtuen tavallisesti onnistu, mutta suorat UDP-yhteydet voidaan kuitenkin saattaa toimimaan tietyin edellytyksin. Näitä menetelmiä, sekä niihin liittyviä vaatimuksia pohditaan

lisää myöhemmin kohdassa 8.1.



Kuva 4.2: *Wireshark-analysointityökalu toiminnassa*

Lisäksi huomattiin, että kahden suomessa sijaitsevan mobiiliverkoissa kiinni olevien pelaajien välille ei onnistuttu luomaan minkäänlaista vertaisverkkoyhteyttä, sillä myös UDP-paketit kierrätettiin aiemmin löydettyjen palvelinten kautta. On siis varsin mahdollista, että pelkästään mobiiliyhteyksiä käyttävien pelaajien välille on vaikeampi luoda vertaisverkkoa, kuin esimerkiksi WLAN:n kautta yhdistäneiden pelaajien välille. Kuvassa 4.2 esitetään Wireshark-ohjelma toiminnassa näiden kahden pelaajan välillä ja havaitaan, että sekä TCP että UDP paketit kulkevat osoitteen 173.194.71.127 kautta.

4.3 Ongelmakohteet

Google Play Games Services SDK:n moninpelirajapinnan käyttöön liittyy kuitenkin Hyperkani Oy -yrityksen näkökulmasta monia ongelmia. Eräs ongelmallinen vaatimus on palvelussa tarvittava tunnistautuminen verkkoyhteisöpalvelu Google+-tilillä [24], sillä vaikka rajapinta tukeekin myös iOS-käyttöjärjestelmiä, on Google+-tilien käyttäminen varsin harvinaista iOS-käyttäjien keskuudessa. Lisäksi Google Play-palvelun saatavuus asettaa sitä hyödyntäville tuotteille myös omat rajoituksensa, sillä esimerkiksi Kiinassa palvelun käyttö on estetty [28].

Kolmas ongelmakohta liittyy tarjotun rajapinnan suljettuun lähdekoodiin, sillä palvelun käytön aikana vastaan on tullut tilanteita, jotka olisivat ratkenneet pienillä

muutoksilla muutoin avoimen libjinglen lähdekoodiin. Esimerkiksi uusien pelaajien liittyminen jo olemassa olevaan huoneeseen on nykyisen rajapinnan kautta mahdotonta [24], vaikka kyseiselle toiminnolle olisikin tarvetta aiemman pelaajan lähtiessä huoneesta neljän pelaajan moninpeleissä. Suljetusta kirjastosta johtuen rajapinnan avulla toteutetut pelit ovat myöskin täysin riippuvaisia Google Play Games Services -palvelun toiminnasta sekä siihen kohdistuvista muutoksista.

Lisäksi käytössä on havaittu tilanteita, joissa unreliable-yhteydet (UDP-protokolla) joidenkin tahojen välillä saattavat arvaamattomasti katketa. Tilanteesta ei kuitenkaan aiheudu virhetapahtumaa Google Play -komponentin puolesta, joten tähän tapahtumaan ei voida varautua ilman erillistä logiikkaa katkeamisen tunnistamiseksi. Toisaalta komponentti ei myöskään mahdollista yhteyksien uudelleenluomista käyttäjien puolelta, sillä huoneiden luomisen edellytyksenä on onnistuneesti rakennettu vertaisverkko kaikkien tahojen välillä molemmilla eri protokollilla, eivätkä kehittäjät pysty vaikuttamaan niissä havaittuihin toimintahäiriöihin [24].

4.4 Vaatimukset ratkaisulle

Mainituista ongelmakohdista sekä palvelun tarpeellisuudesta johtuen Hyperkani Oy on päättänyt tutkia reaaliaikaisten moninpelien mahdollisuutta ilman Google Play Games Services SDK:n tarjoamaa toiminnallisuutta.

Järjestelmän tulee luonnollisesti toimia sekä Android- että iOS-käyttöjärjestelmillä eikä se saa olla riippuvainen Google Play -SDK:n toiminnoista aiemmin mainittujen ongelmien välttämiseksi. Järjestelmän tulee olla skaalautuva ja sen pitää kyetä palvelemaan parhaimmillaan kymmeniä tuhansia yhtäaikaisia pelaajia. Avoimen lähdekoodin kirjastojen käyttö on suotavaa lisenssiehdot huomioon ottaen siksi, että järjestelmä olisi muokattavissa yrityksen toiveiden mukaiseksi. Lisäksi järjestelmän tulee olla myöhemmin laajennettavissa sisältämään esimerkiksi pelaajaprofiilit sekä profiilien väliset kaverisuhteet.

Lisäksi Luvussa 4.2 käsitellyn analysointihistorian perustuessa tavalliseksi luonnehdittavaan pelikokemukseen, voidaan tämän historian perusteella arvioida ratkaisulle asetettavia tiedonsiirtoa koskevia vaatimuksia. Tällaisia vaatimuksia ovat esimerkiksi käsiteltävien pakettien määrä sekuntia kohden sekä pakettien sisältämän sisällön suuruus. Taulukkoon 4.1 on koottu näiden historiatietojen perusteella käsiteltyjen pakettien määrä sekä pakettien sisältöjen suuruus kahden pelaajan välillä yhden sekunnin aikana. Neljän pelaajan välisessä pelissä määrät kolminkertaistuvat.

Taulukko 4.1: *Keskimääräiset tiedonsiirtomäärät sekuntia kohden verkon analysointityökalun tuottaman datan perusteella*

	Lähetetty	**Vastaanotettu**
TCP-pakettien lukumäärä	4 kpl	4 kpl
UDP-pakettien lukumäärä	7 kpl	3 kpl
Siirretyn datan määrä TCP-yhteydellä	136 tavua	136 tavua
Siirretyn datan määrä UDP-yhteydellä	238 tavua	102 tavua

Taulukon 4.1 perusteella arvioidaan korvaavan järjestelmän vaatimuksiksi, että sen pitää pystyä käsittelemään kahden pelaajan välillä ainakin vastaava määrä liikennettä ilman ongelmia. Vaatimukset täytettyään uuden järjestelmän olisi mahdollista korvata Google Play Game Services -moninpelirajapinta ilman verkkoliikenteeseen vaikuttavia muutoksia Bomber Friends -pelissä.

4.5 Vaihtoehtoisia ratkaisumenetelmiä

Ratkaisuvaatimusten selvittyä käsitellään sekä joko hyväksytään tai hylätään erilaisia vaihtoehtoisia ratkaisumenetelmiä.

4.5.1 Asiakas-palvelin-malli ja pelipalvelin

Ensimmäinen vaihtoehto verkkopelijärjestelmälle on asiakas-palvelin-mallin hyödyntäminen sekä erillisen pelipalvelimen käyttöönottaminen. Tämä ratkaisu mahdollistaisi käynnissä olevien pelien ylläpidon sekä pelien tilatietojen säilyttämisen palvelimella, joka vastaisi saapuneiden tietojen laillisuuden ja oikeellisuuden tarkistamisesta.

Erillinen pelipalvelin vaatii kuitenkin jatkuvaa ylläpitoa, ja vaikka järjestelmä toimisikin luotettavasti sen käyttöönoton alkuvaiheessa, voi pelaajamäärien kasvaminen tehdä sen toiminnasta epävakaata niin yhteyksien kuin suorituskyvynkin kannalta. Mikäli pelaajat sijaitsevat esimerkiksi kokonaan eri mantereilla, tuottaa tietojen kierrättäminen palvelimen kautta myös ylimääräistä viivettä sovellusten väliseen tiedonsiirtoon. Lisäksi pelien toimintojen ollessa riippuvaisia yksittäisestä pelipalvelimesta, voivat esimerkiksi sähkökatkot tai palvelimeen kohdistuneet palvelunestohyökkäykset estää pelien pelaamisen kokonaan.

Laajentamalla ratkaisua kattamaan useampi pelipalvelin ja hajauttamalla nämä palvelimet eri mantereille saataisiin lisää suorituskykyä sekä vastattaisiin pelaajamäärien kasvamisen aiheuttamaan ongelmaan. Tällöin maantieteellisesti samalla suunnalla sijaitsevat käyttäjät pelaisivat aina keskenään ja viiveet pysyisivät matalina. Järjestelmän toiminta ei olisi myöskään riippuvainen yksittäisen palvelimen toiminnasta.

Useamman pelipalvelimen alustaminen sekä ylläpitäminen on kuitenkin resursseja tuhlaava ratkaisu, eikä se sellaisenaankaan ratkaise mahdollisesti tietyille alueille keskittyviä pelaajamääriin kohdistuvia radikaaleja muutoksia. Ratkaisu olisi kuitenkin teknisesti toteutettavissa, mikäli palvelimet esimerkiksi viestivät ja tasaavat kuormaa keskenään käyttöasteiden mukaan. Toisaalta tällöin on jälleen mahdollista, että pelaajien väliset viiveet kasvavat.

4.5.2 Avoimet vertaisverkkokirjastot

Vaihtoehtoisesti reaaliaikainen moninpelijärjestelmä voidaan toteuttaa nykyisen käytössä olevan Google Play Games Services SDK-kirjaston tavoin vertaisverkkotekniikalla, jossa samaa peliä keskenään pelaavat pelaajat muodostavat vertaisverkon sisällön vaihtamista varten. Tämä ratkaisu sisältää yhden tai useamman matchmaking-palvelimen, joille pelaajat yhdistävät muiden pelaajien etsimistä varten ennen vertaisverkon muodostamista. Tällaiset palvelimet eivät kuitenkaan joudu suuren rasituksen alle, sillä pelaajat ovat niihin yhteydessä ainoastaan peliseuraa etsiessään, eivät peliä pelatessaan. Muiden pelaajien löydyttyä voidaan yhteys matchmaking-palvelimelle katkaista.

Vertaisverkon luomiseen on saatavilla erillisiä ICE-protokollaa noudattavia useita eri käyttöjärjestelmiä tukevia avoimen lähdekoodin vertaisverkkokirjastoja, kuten esimerkiksi Libjingle [26] ja Libnice [36]. Libjingle on ajonaikaisten tulosteiden (log) perusteella käytössä myös Google Play Game Services SDK:ssa sekä esimerkiksi jo käytöstä poistuneessa Google Talk -pikaviestiohjelmassa, ja sen yhteyksien muodostaminen eli signalointi perustuu pikaviestintään sekä läsnäolon seurantaan kehitettyyn XMPP-protokollaan [21]. Libnice on sen sijaan on suppeampi sekä yksinkertaistetumpi vastine Libjinglelle ja se sisältää ainoastaan toteutukset eri ICE-standardien noudattamiselle vertaisverkkojen muodostamista varten. Taulukossa 4.2 vertaillaan näiden kahden kirjaston tämän työn kannalta oleellisimpia eroavaisuuksia.

Jingle-protokollalla viitataan XMPP-pikaviestijärjestelmän laajennokseen [39], jos-

Taulukko 4.2: *Libjinglen ja Libnicen oleellisimmat eroavaisuudet [26][36]*

	Libjingle	**Libnice**
ICE-toteutus	Toteuttaa *jingle*-standardit (XEP-166, XEP-167 XEP-176, XEP-177)	Sisältää usean eri standardin toteutuksen (ICE, STUN, TURN)
Signalointi	XMPP-protokollalla	Ei sisälly kirjastoon
Kehitystyö	Lopetettu kesällä 2013	Jatkuu toistaiseksi
Rajapinta	Puutteellinen ja epäselvä dokumentointi	Selkeä dokumentointi sekä kattavat esimerkit
Muuta	Käytössä useissa Googlen tuotteissa, nykyään WebRTC:n pohjana	Toimiva ja aktiivinen sähköpostilista kysymyksille ja kommentoinnille
Lisenssi	Copyright 2004–2007 Google Inc.	LGPL ja MPL-1.1

sa samaisen pikaviestijärjestelmän käyttäjät voivat tarvittaessa muodostaa keskenään vertaisverkkoja esimerkiksi ääni- ja videopuheluita sekä tiedostojen jakamista varten. Googlen vuonna 2005 julkaisema Libjingle toteuttaa tämän standardoidun protokollan ja sitä hyödynnetään Google Talk -pikaviestinpalvelun sekä Google Play Games Services SDK:n lisäksi esimerkiksi Google Chrome ja Mozilla Firefox -selaimissa [30].

Vaikka Libjingle-kirjasto sisältääkin kattavan määrän eri ominaisuuksia, sen kehitys on sellaisenaan lopetettu kesällä 2013 ja se on sisällytetty osaksi WebRTC-ohjelmointirajapintaa, jossa se pyrkii mahdollistamaan vertaisverkkojen muodostamisen kahden selaimen välille ilman erillisiä selainlaajennoksia [30][29, s. 309-319]. Tästä johtuen Libjinglen dokumentointi sekä siihen liittyvät esimerkit ovat sellaisenaan toistaiseksi varsin rajallisia ja sen hyödyntäminen kehitystyössä on hankalaa. Lisäksi verkkopelijärjestelmän prototypointia silmälläpitäen XMPP-protokollan noudattaminen matchmaking-palvelimella on ainakin kehitystyön aikana tarpeetonta. Toisaalta lopullisessa järjestelmässä XMPP-pohjainen järjestelmä sisältäisi valmiin toteutuksen esimerkiksi kaverisuhteiden luomiselle ja pikaviestittelylle verkkopelaamisen ohessa.

Matalan tason GLib-järjestelmäkirjastokokoelmaan pohjautuva Libnice on puolestaan selkeästi suppeampi kirjasto kuin Libjingle ja se on tarkoitettu ainoastaan vertaisverkkoyhteyksien muodostamiseen ICE-protokollalla [36]. Dokumentaatiosta löytyvät havainnollistavat esimerkit tekevät siitä helppokäyttöisen sekä tarvittaessa vai-

vattomasti omaan käyttöön muokattavan vertaisverkkokirjaston. Signaloinnin puutteellisuus katsotaan prototypointivaiheessa eduksi, sillä työhön olennaisesti kuuluva matchmaking-palvelin saadaan toimimaan myös vertaisverkkojen signalointipalvelimena.

Ainoat Libnicen käytössä esiintyneet ongelmat koskevat sen toimintakuntoon saattamista Android-järjestelmälle, sillä vaikka itse lähdekoodien taataankin olevan käyttöjärjestelmäriippumattomia, ei sille löytynyt valmista Makefile-käännöstiedostoa tai muuta valmista työkalua kirjaston kääntämiseksi. Libnicen aktiivisesti ylläpidetty sähköpostilista [37] tuotti kuitenkin vastauksia kääntämisen aikana heränneisiin kysymyksiin, ja Android-käännöksen onnistuttua kirjaston käyttöönotto oli suoraviivaista ja selkeää.

Toteutettaessa reaaliaikaiset moninpelit vertaisverkkotekniikalla, kehittäjien tulee huomioida pelin tilatietojen ylläpitäminen sekä päivittäminen verkon tahojen välillä siten, että pelin toiminta jatkuu, vaikka yksittäinen pelaaja lähtisikin verkosta. Lisäksi kehittäjien on huomioitava tilanteet sekä toimintamenetelmät vertaisverkon luomisen epäonnistumiselle esimerkiksi osoitteenmuunnosten tai palomuurien takia.

Vertaisverkkojen useista eduksi katsottavista ominaisuuksista johtuen reaaliaikainen moninpelirajapinta toteutetaan kuitenkin vertaisverkkotekniikalla Libnice-kirjastoa hyödyntäen. Koska reaaliaikaisen moninpelirajapinnan tulee tarvittaessa toimia myös iOS-käyttöjärjestelmällä, se toteutetaan luvussa 2.3 esiteltyä natiivikoodia hyödyntäen esimerkinmukaisena jaettuna kirjastona, jolloin se on sellaisenaan käännettävissä myös iOS-järjestelmälle.

5. REAALIAIKAISEN MONINPELIRAJAPINNAN PROTOTYYPPI

Käytössä olevaan komponenttiin liittyvistä ongelmista johdettujen ratkaisuvaatimusten perusteella toteutettiin prototyyppi reaaliaikaisesta moninpelirajapinnasta. Järjestelmä mahdollistaa peliseuran etsimisen julkisen matchmaking-palvelimen avulla sekä vertaisverkon luomisen ja sisällön välittämisen kahden pelaajan välillä. Prototyyppiä ei kehitetty lopulliseksi tuotantokäyttöön soveltuvaksi ratkaisuksi kohdassa 4.2 kuvattujen tahojen välisten säännöllisten yhteysongelmien vuoksi.

Prototyyppi jakautuu seuraaviin osakokonaisuuksiin: laitteistolle tarjottava rajapinta, matchmaking-palvelin muiden pelaajien etsimistä varten sekä relay-palvelin yhteyksien muodostamiseen vertaisverkon luomisen epäonnistuessa. Tässä luvussa esitellään järjestelmän toimintaperiaatteet.

5.1 Laitteistolle tarjottava rajapinta

Käyttäjälle tarjottava moninpelirajapinta on kehitetty C++-kielellä tämän ollessa Hyperkani Oy:n pääsääntöisesti käytettävä kehityskieli kohdekäyttöjärjestelmäriippumattomien (Android, iOS) osioiden toteuttamiseen. Komponentti käyttää hyödykseen edellisessä luvussa esiteltyä Libnice-vertaisverkkorajapintaa, joka puolestaan vaatii toimiakseen GLib-kirjastokokoelman [36]. Rajapinta on yksinkertainen ja tarkoitettu ainoastaan havainnollistamaan prototyypin toimintaa niiltä osin kuin on tämän työn kannalta tarpeellista. Lisäksi prototyypin rajapintaa on myös helppo tarvittaessa laajentaa kattavammaksi järjestelmäksi yrityksen toiveiden mukaisesti.

Komponentin sekä vastaavasti sitä käyttävän ohjelman kääntäminen onnistuu Androidin omalla NDK-työkalukokoelmalla. Tämän kokoelman sisältämät kääntäjät mahdollistavat natiivikoodin kuten esimerkiksi C++:n kääntämisen Android-järjestelmän ymmärtämään muotoon luvun 2.3 mukaisesti. Tavallisimmissa mobiilisovelluksissa natiivikoodia hyödynnetään esimerkiksi suorituskykykriittisten osioiden toteuttamiseen, mutta useat peliohjelmointikehykset, kuten esimerkiksi Hyperkani Oy:n pääasiallisesti hyödyntämä Cocos2d-X [8], on toteutettu C++ kielellä sekä suorituskyvyn että usean eri kohdekäyttöjärjestelmätuen takia.

```
9
10      // alustusfunktio, tulee kutsua erillisessä säikeessä
11      static void initMultiplay();
12      // tarvittaessa luo ja palauttaa käytössä olevan Multiplay-olion,
13      // initMultiplay()-funktiota tulee kutsua ennen tämän funktion kutsumista
14      static Multiplay *getInstance();
15
16      // nollaa komponentin tilan, katkaisee olemassaolevat yhteydet
17      void resetComponent();
18
19      // yhdistää halutulle matchmaking-palvelimelle
20      void connectToServer( MultiplayInterface *listener,
21          std::string mpServer, int port,
22          std::string relayServer, int relayPort );
23      // katkaisee yhteyden matchmaking-palvelimelta
24      void disconnectFromServer();
25
26      // lähettää dataa muodostettuun vertaisverkkoon
27      void sendData( const char *data );
28      // katkaisee vertaisverkkoyhteydet
29      void disconnectP2P();
30
```

Kuva 5.1: *Osa moninpelikomponentin julkista rajapintaa*

Yksinkertaisuutensa vuoksi prototyypin perustoiminnot jakautuvat kahteen eri luokkaan, joista ensimmäinen vastaa eri yhteyksien luomisesta sekä ylläpidosta, kun taas toinen toimii takaisinkutsurajapintana käyttäjälle komponentin tilan sekä viestien saapumisen seurantaan. Vaikka komponentin rajapinta onkin esitelty luokkana, tämä luokka on tyyppiä *singleton* eli siitä on mahdollista luoda ainoastaan yksi olio komponentin käyttämistä varten. Osa moninpelikomponentin julkisesta rajapinnasta on esitetty kuvassa 5.1 ja käyttäjän toteuttamat takaisinkutsufunktiot puolestaan on esitetty kuvassa 5.2. Lisäksi komponentin eri tilatiedot on esitetty kuvassa 5.3.

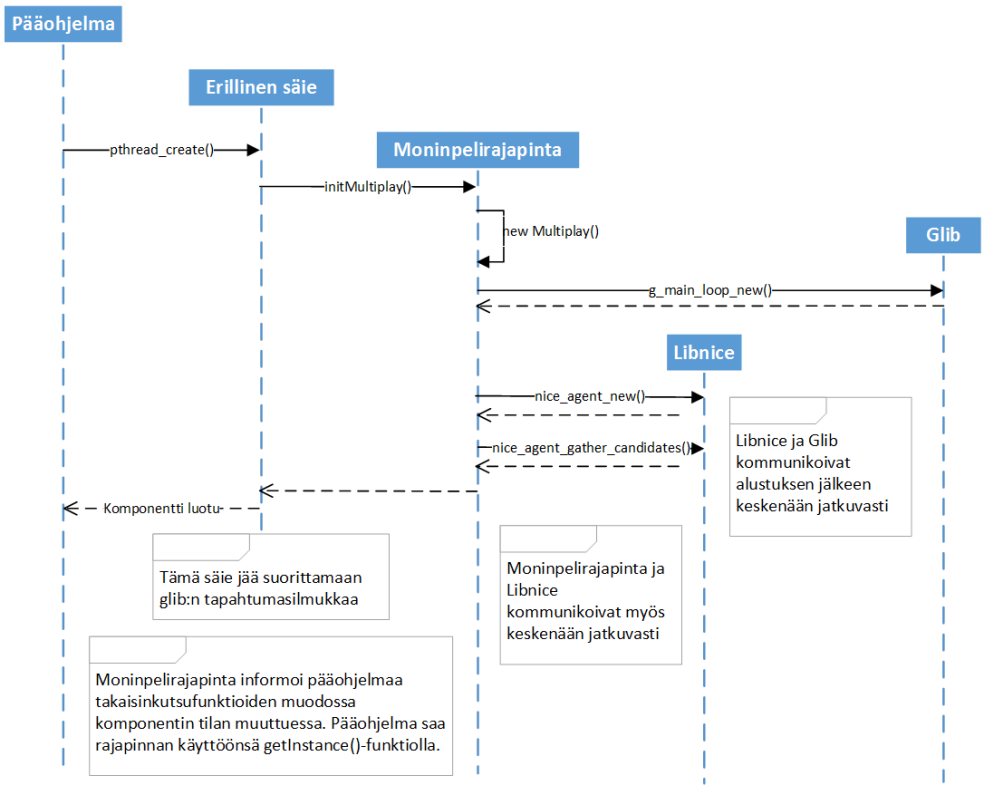
```
35  // kuuntelija-rajapinta
36  class MultiplayInterface
37  {
38  public:
39      // takaisinkutsufunktio, kutsutaan komponentin tilan muuttuessa
40      virtual void MultiplayComponentStateChanged( MultiplayComponentState newState ) = 0;
41      // takaisinkutsufunktio, kysyy käyttäjältä muodostetaanko P2P-yhteys
42      virtual bool MultiplayComponentRequestingP2PConnection() = 0;
43      // takaisinkutsufunktio, vertaisverkon kautta on vastaanotettu dataa
44      virtual void dataReceived( const char *data ) = 0;
45  };
```

Kuva 5.2: *Takaisinkutsuluokka, jonka avulla käyttäjä saa tiedon komponentin eri tapahtumista*

```
21      // komponentin tilatiedot
22      enum MultiplayComponentState
23      {
24          MP_STATE_INITIALIZED,
25          MP_STATE_CONNECTING_TO_SERVER,
26          MP_STATE_CONNECTED_TO_SERVER,
27          MP_STATE_REQUESTING_P2P_CONNECTION,
28          MP_STATE_CONNECTING_TO_P2P,
29          MP_STATE_CONNECTED_TO_P2P,
30          MP_STATE_CONNECTING_TO_RELAY,
31          MP_STATE_CONNECTED_TO_RELAY_AND_WAITING_OPPONENT,
32          MP_STATE_CONNECTED_TO_RELAY,
33          MP_STATE_UNKNOWN
34      };
```

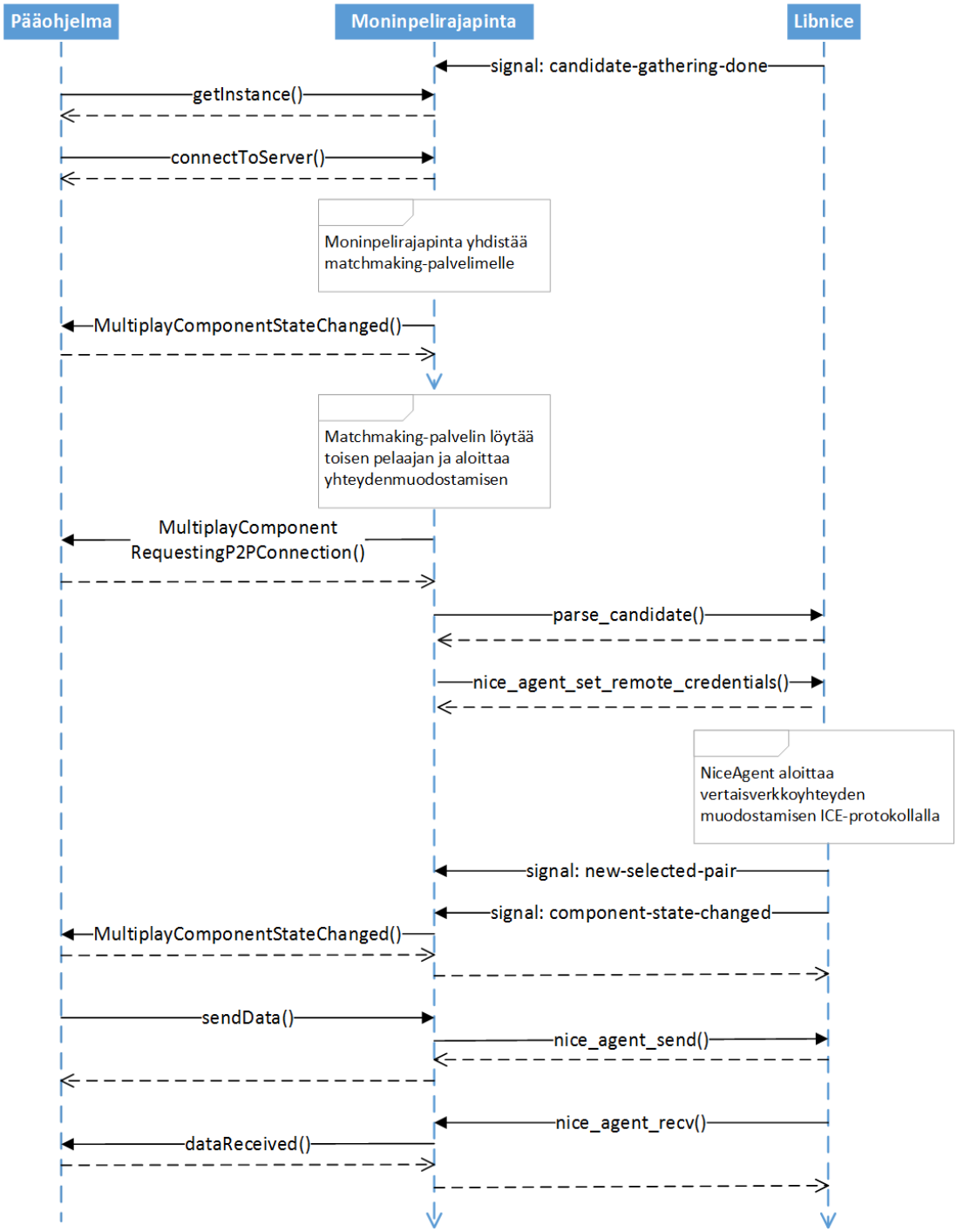
Kuva 5.3: *Moninpelikomponentin mahdolliset tilat*

Komponentin käyttö aloitetaan kutsumalla initMultiplay-funktiota erillisessä säikeessä (thread). Tässä funktiossa alustetaan Libnicen vaatima GLib-kirjasto käyttöä varten ja luodaan esimerkiksi tarvittavat ajastimet. Erillisessä säikeessä alustusfunktiota tulee kutsua siksi, että alustuksen yhteydessä aloitetaan g_main_loop-päätapahtumasilmukan suoritus, minkä johdosta tästä funktiosta ei milloinkaan palata ohjelman elinkaaren aikana. Sekvenssikaavio 5.4 havainnollistaa moninpelikomponentin käyttöönottoa sekä sen toimintaperiaatteita.



Kuva 5.4: *Sekvenssikaavio rajapinnan alustamisen toimintaperiaatteesta*

Tämän jälkeen komponentti sekä luodaan että palautetaan kutsumalla getInstance-funktiota. Tämän funktion sekä piilotetun rakentajan avulla myös varmistetaan, ettei komponentista ilmennetä useita samanaikaisia olioita prototyyppiin liittyvien rajoitusten vuoksi.



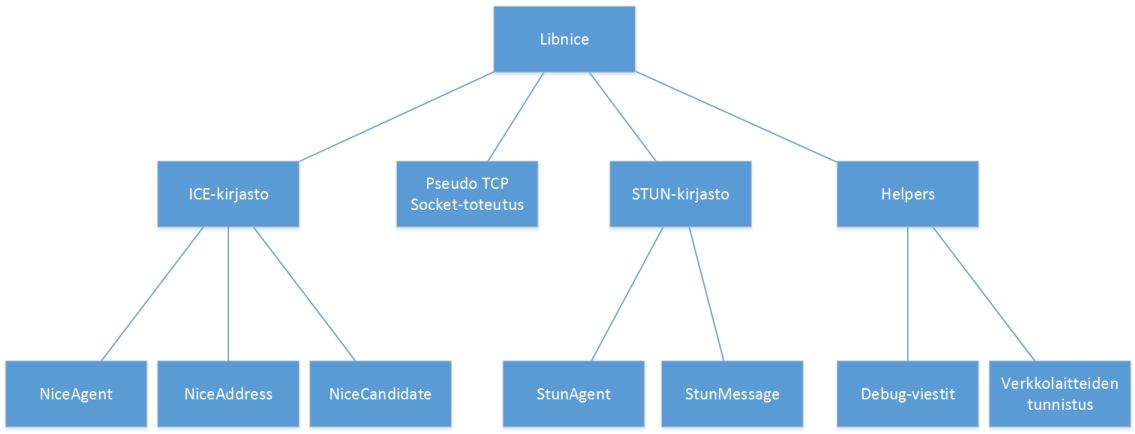
Kuva 5.5: *Sekvenssikaavio moninpelikomponentin toiminnasta*

Seuraavaksi käyttäjä yhdistää matchmaking-palvelimelle peliseuran etsimistä varten connectToServer-funktion avulla. Peliseuran löydyttyä käyttäjältä tiedustellaan kuvan 5.2 takaisinkutsufunktioiden avulla muodostetaanko yhteys löydetyn pelaajan kanssa. Palaute yhteydenmuodostuksen onnistumisesta saadaan vastaavasti MultiplayComponentStateChanged-funktion kautta. Molempien pelaajien tulee hyväksyä yhteyspyyntö vertaisverkon muodostamisen aloittamiseksi. Yhteyden muodostuttua pelaaja voi lähettää sekä vastaanottaa dataa sendData- ja dataReceived-funktioiden avulla. Kuvan 5.5 sekvenssikaavio havainnollistaa komponentin perustoimintojen käyttämistä.

Komponentin käyttö päätetään sulkemalla mahdollisesti muodostettu yhteys disconnectP2P-funktiolla tai vaihtoehtoisesti nollaamalla koko komponentin tila resetComponent-funktiolla. Verkon toinen taho vastaanottaa tiedon yhteyden katkeamisesta takaisinkutsurajapinnan kautta. Komponentin toiminta nollataan resetComponent-funktiolla seuraavia yhteyksiä varten.

5.1.1 Libnice-kirjaston toiminta rajapinnan yhteydessä

Libnice-kirjaston toiminta käydään läpi lähteen [35] mukaisesti. Moninpelirajapinnan pohjana toimiva Libnice-kirjasto jakautuu kuvan 5.6 mukaisesti neljään eri osa-alueeseen: ICE-kirjastoon, STUN-kirjastoon, Pseudo TCP Socket-toteutukseen sekä erilaisiin aputyökaluihin.



Kuva 5.6: *Libnice-kirjaston arkkitehtuuri*

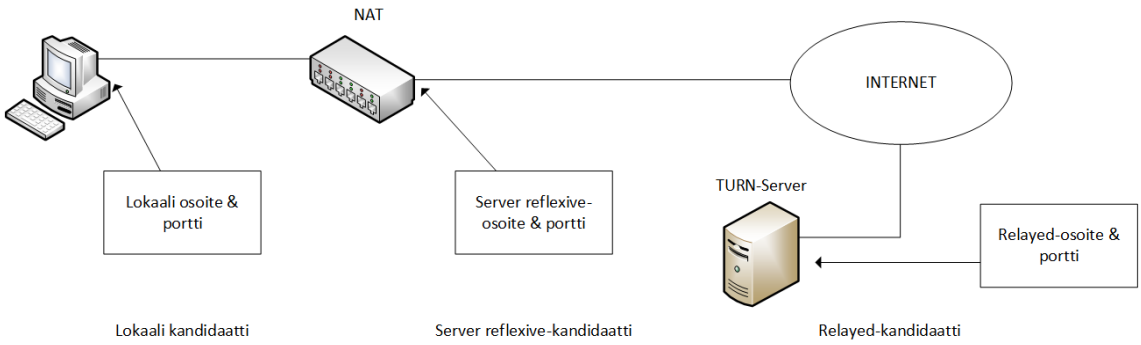
Libnice-kirjaston käyttö aloitetaan luomalla GObject-tyyppinen NiceAgent-instanssi, jonka tehtävänä on sekä alustaa että ylläpitää Libnicen toimintaa sen elinkaaren

aikana. NiceAgent toimii siten rajapintana muihin kirjaston toimintoihin ja siksi lähes kaikki Libnice-kirjaston julkiset funktiot vaativat viittauksen olemassaolevaan NiceAgent-olioon. On huomattava, että NiceAgent-rakentajalle annetaan parametrinä osoitin komponenttia alustettaessa luotuun GLib-kirjaston g_main_loop-tapahtumasilmukkaan, sillä agentin ajastimet ja muut perustoiminnot ovat tästä tapahtumasilmukasta riippuvaisia.

Agentin luomisen jälkeen alustetaan ainakin yksi virta tiedonsiirtoa varten sekä yhdistetään GLib-kirjastoon sisällytetyn signal-slot-moduulin takaisinkutsufunktiot. Näiden alustusten jälkeen voidaan aloittaa vertaisverkon yhteyskandidaattien (NiceCandidate) etsiminen STUN-kirjaston avulla.

Ennen vertaisverkkoyhteyden muodostamista ICE-protokollan toteutukseen kuuluu oleellisena osana yhteyskandidaattien etsiminen ja löytäminen. Kandidaateilla tarkoitetaan tässä yhteydessä priorisoituja osoite-portti-pareja, joita vastapuoli voisi mahdollisesti hyödyntää muodostaessaan yhteyttä kyseiseen tahoon. Osoitteen, portin ja prioriteetin lisäksi yksittäisestä kandidaatista selviää ainakin tämän kandidaatin protokolla sekä kandidaatin tyyppi. [49]

Yhteyskandidaatit jaetaan tavallisesti kolmeen eri tyyppiin: lokaalit kandidaatit, server reflexive-kandidaatit ja relayed-kandidaatit [49]. Lokaalit kandidaatit ovat laitteen omilta verkkolaitteilta saatuja lokaaleita osoitteita (esim. 10.0.0.25), jotka ovat sellaisten tahojen saavutettavissa, jotka sijaitsevat samassa lähiverkossa tämän laitteen kanssa. Server reflexive-kandidaatit puolestaan vastaavat STUN-protokollalla selvitettyä verkkolaitteen osoitetta, joka sijaitsee tämän tahon sekä julkisen internetin välissä. Lisäksi relayed-kandidaatilla tarkoitetaan mahdollisesti käytössä olevan relay-palvelimen osoitetta. Kuva 5.7 havainnollistaa näiden eri kandidaattityyppien tarkoitusta.



Kuva 5.7: *Eri kandidaattityypit*

Yhteyskandidaattien selvittyä Libnice jää odottamaan vastakkaisen tahon tuottamaa listaa vastaavista yhteyskandidaateistaan yhteydenmuodostusta varten. Tahon kandidaattilistan saatuaan se pyrkii muodostamaan yhteyden näiden saatujen kandidaattitietojen perusteella korkeimman prioriteetin omaavasta kandidaatista aloittaen. Tavallisimmin yhteys saadaan muodostettua viimeistään alhaisimman prioriteetin omaavan relayed-kandidaatin tietojen perusteella, vaikkakaan tällöin kyse ei ole enää suorasta yhteydestä tahojen välillä.

ICE- ja STUN-kirjastojen lisäksi Libnice sisältää Pseudo TCP Socket-objektin toteutuksen. Pseudo TCP Socket-objektilla tarkoitetaan UDP-protokollaa hyödyntävää yhteysmuotoa, joka toteuttaa osan TCP-pinosta luotettavan tiedonsiirron takaamiseksi. Pseudo TCP Socket-objekti on käyttökelpoinen esimerkiksi tilanteissa, joissa vertaisverkko tahojen välille onnistutaan muodostamaan ainoastaan UDP-protokollalla.

5.1.2 Yhteys matchmaking-palvelimeen

Yhteys matchmaking-palvelimeen muodostetaan TCP-protokollalla erillisessä tarkoitusta varten luodussa säikeessä. Yhteyden luonnin onnistuessa käyttäjän välittämä matchmaking-parametri lähetetään palvelimelle kriteeriksi parien muodostamiselle. Tämän jälkeen käyttäjä odottaa takaisinkutsurajapinnan kautta välitettävää signaalia tiedoksi siitä, että sopiva pari on löytynyt.

Peliseuran löydyttyä nämä parit vaihtavat keskenään omia löytyneitä kandidaattejaan matchmaking-palvelimen kautta. Kandidaattien vaihtamisen jälkeen yhteys matchmaking-palvelimeen suljetaan ja aloitetaan vertaisverkkoyhteyden muodostaminen.

5.1.3 Vertaisverkkoyhteyden muodostus

Kun molemmat tahot ovat vastaanottaneet listan vastapuolen kandidaateista vertaisverkkoyhteyden muodostamista varten, yritetään muodostaa yhteys vastapuolen korkeimman prioriteetin kandidaatin tietojen perusteella. Mikäli yhteydenmuodostus tähän kandidaattiin epäonnistuu, yritetään uudelleen seuraavaksi tärkeimmän kandidaatin kanssa. Tätä toistetaan niin kauan, kunnes saadaan ainakin yksi yhteys tahojen välille aukeamaan. Moninpelirajapinnan yhteydessä listan lopusta löytyy relayed-kandidaatti, jonka avulla yhteyden muodostus mitä todennäköisimmin viimeistään onnistuu. Mikäli yhteys ei kuitenkaan relay-palvelimenkaan kautta muo-

dostu esimerkiksi palvelimeen kohdistuneen sähkökatkon takia, ei näitä kahta tahoa saada tällä kerralla yhdistettyä keskenään. [49]

5.1.4 Kommunikointi vertaisverkon välityksellä

Yhteyden muodostamisen onnistuttua moninpelirajapinnan käyttäjä voi lähettää dataa vastapuolelle sendData-funktion avulla. Datan vastaanottaminen tapahtuu dataReceived-takaisinkutsufunktion avulla. Reaaliaikainen moninpelirajapinta ylläpitää avattua yhteyttä sekä tarkastelee jatkuvasti sen tilaa yhteysongelmien varalta. Yhteyden tilan muuttuessa käyttäjä saa tästä tiedon takaisinkutsurajapinnan kautta.

5.1.5 Vertaisverkkoyhteyden sulkeminen

Yhteys suljetaan disconnectP2P-funktiolla pelaajan lopettaessa pelaamisen. Vertaisverkon toiselle taholle tiedotetaan yhteyden sulkeutumisesta takaisinkutsurajapinnan kautta. Mikäli pelaaja haluaa pelata uudelleen, muodostetaan uusi yhteys matchmaking-palvelimelle peliseuran etsimistä varten.

5.1.6 Kehittämisessä kohdattuja ongelmia

Reaaliaikaisen moninpelirajapinnan kehitystyön aikana kohdattiin joitakin ongelmia. Selkeästi suurin yksittäinen ongelma oli kehitystyön aloittaminen kääntämällä sekä GLib että Libnice staattisiksi Android-moduuleiksi. Tämä edellytti useamman apukirjaston kääntämistä erikseen tietyssä järjestyksessä ennen GLibin kääntämistä. Tällaisia apukirjastoja olivat esimerkiksi *libffi*, *libgmodule*, *libgobject*, *libgstbase*, *libiconv*, ja *libxml2*. Glib-kirjastoa puolestaan tarvittiin Libnice-kirjaston kääntämiseen.

Toinen merkittävä ongelma liittyi vertaisverkkojen yhteydenmuodostamiseen, sillä mobiiliverkkoon yhdistäneitä laitteita ei sellaisenaan saatu muodostamaan suoraa yhteyttä keskenään. Samassa lähiverkossa sijaitsevat laitteet kuitenkin onnistuivat suoran yhteyden muodostamisessa. Ongelma johtuu yhteyksien välissä käytettävissä osoitteenmuunnoksista ja sen ratkaisuksi on kehitelty erilaisia menetelmiä, joita käsitellään myöhemmin luvussa 8.1.

5.2 Matchmaking-palvelin

Matchmaking-palvelimen tehtävänä on muodostaa pareja muita käyttäjiä etsivistä pelaajista ennalta määrättyjen kriteerien mukaisesti. Palvelinohjelmisto on kehitetty C++-kielellä Qt-kirjaston avulla ja se sisältää myös yksinkertaisen TURN-palvelimen toteutuksen yhteyksien muodostamiseen kahden osoitteenmuunnoksen takana olevan laitteen välille. Moninpelirajapinnan tavoin palvelimen prototyyppi on yksinkertaistettu toteutus palvelimen lopullisesta julkaisukäyttöön tarkoitetusta versiosta, mutta sisältää kuitenkin kaikki vaadittavat toiminnot palvelimen suorituskyvyn sekä tarkoitukseen soveltuvuuden arviointia varten.

5.2.1 Matchmaking-palvelimen toimintaperiaate

Matchmaking-palvelimen käyttäminen alkaa asiakasohjelmiston yhteyden muodostamisella, jonka aikana asiakasohjelman tulee autentikoitua palvelimelle. Yhteyden muodostuttua palvelin pyytää asiakasta toimittamaan erilaisia pelaajien etsimiseen vaikuttavia parametrejä. Nämä parametrit sisältävät tietoja esimerkiksi pelattavasta pelistä, pelityypistä, asiakkaan sijainnista ja muusta oleellisesta pelaajien hakemiseen mahdollisesti vaikuttavista ominaisuuksista. Vaadittujen parametrien vastaanottamisen jälkeen aloitetaan hakuprosessi.

Matchmaking-palvelimen hakuprosessiin on kehitetty useita erilaisia algoritmejä, mutta mallin yksinkertaistuksesta johtuen prototyyppiin toteutettiin kuitenkin ainoastaan naiivi hakualgoritmi, jossa pelaajien väliset yhteydet muodostetaan FIFO-jonokurin perusteella, jossa aiemmin palvelimelle yhdistäneelle pelaajalle pyritään etsimään pelikaveri ensimmäisenä.

5.2.2 Matchmaking-palvelimeen liittyviä ongelmia

Toteutettua prototyyppiä testattiin varsin rajallisesti ainoastaan muutamilla yhtäaikaisilla käyttäjillä, eikä sen toimivuutta tuotannossa voida siksi arvoida realistisesti. Testausprosessin aikana tiedostettiin kuitenkin useita sellaisia palvelimen käyttöön liittyviä ongelmia, joihin tulee varautua ennen matchmaking-palvelimen varsinaista toteutusta.

Merkittävimmät palvelimen käytössä esiintyvät ongelmat liittyvät palvelimelle asetettaviin saavutettavuusvaatimuksiin ja nämä ovat kiinteästi yhteydessä palvelimen totetuksessa hyödynnettyyn asiakas-palvelin-arkkitehtuuriin. Kuten jo kohdassa 4.5.1 todettiin, yksittäinen järjestelmä on liian haavoittuvainen esimerkiksi pal-

velunestohyökkäyksille tai sähkökatkoille ja toisaalta useamman palvelimen palvelimen pystyttäminen vaatii palvelinten välistä kommunikointia käyttäjien jakamiseksi ja mahdollisten kaatuneiden palvelinten tunnistamiseksi.

5.3 Käytetyt kirjastot

Prototyypin toteutuksessa hyödynnettiin seuraavia julkisesti saatavilla olevia kirjastoja:

- **Libnice**, vertaisverkkorajapinta, joka toteuttaa ICE-protokollan
- **GLib**, kokoelma matalan tason järjestelmäkirjastoja
- **Cocos2d-X**, useaa käyttöjärjestelmää tukeva peliohjelmointikirjasto
- **Qt**, graafisten käyttöliittymien kehitysympäristö

Käytettävien kirjastojen valintaan on vaikuttanut niiden toimivuus sekä Android- että iOS-käyttöjärjestelmillä. Qt-kirjastoa hyödynnettiin ainoastaan matchmaking-palvelimen toteutuksessa.

5.4 Rajapinnan toteutus Libjingle-kirjastolla

Libjingle-kirjaston arkkitehtuuri on suurelta osin vastaavanlainen Libnicen toiminta-periaatteiden kanssa ja tarvittaessa reaaliaikainen moninpelirajapinta voidaan saattaa toimimaan myös Libjingle-kirjastoa hyödyntäen. Esimerkiksi vertaisverkon muodostaminen toteutetaan lähes identtisellä tavalla vaihtamalla yhteyskandidaatteja signalointipalvelimen kautta. Suurimmat erot näiden kahden kirjaston välillä muodostuvat kuitenkin libjinglen vaatimuksesta hyödyntää XMPP-palvelinta vertaisverkon signalointiin, ja toisin kuin tällä hetkellä, tämä muodostettu XMPP-yhteys säilyy avattuna myös vertaisverkkoyhteyden muodostamisen jälkeen [39]. Tämä mahdollistaisi erillisen aula-palvelun toteuttamisen verkkopeleihin, jonka avulla pelaajat voisivat keskustella julkisissa huoneissa keskenään pelikavereita etsiessään.

6. KOMPONENTIN KÄYTTÖÖNOTTO

Reaaliaikaista moninpelirajapintaa ei ainakaan toistaiseksi otettu käyttöön osaksi Hyperkani Oy -yrityksen pelejä. Suurin käyttöönoton estävä ongelma oli vertaisverkon suorien yhteyksien muodostamisen epäonnistuminen liian usein. Ongelmaa selvitettäessä havaittiin lähes kaikkien mobiilioperaattorien hyödyntävän osoitteenmuunnoksia sekä palomuureja mobiililaitteiden edessä, jonka seurauksena kahden laitteen välinen suoran yhteyden muodostaminen ei onnistu. Testatessa kohdassa 4.2 Google Play Game Services -moninpelirajapintaa havaittiin, että usein myös tämän muodostamat yhteydet kulkevat erillisten palvelinten kautta.

Vertaisverkon muodostamisen epäonnistuessa ICE-protokolla määrittelee yhteyden muodostamisen julkisen relay-palvelimen kautta. Tästä johtuen reaaliaikaisen moninpelirajapinnan laajamittainen käyttöönotto vaatisi useamman relay-palvelimen pystyttämistä. Lisäksi palvelinten korkeasta käyttöasteesta johtuen myös yhteyksien tulisi vastata nykypäivän standardeja ja kyetä palvelemaan kymmeniä tuhansia yhtäaikaisia käyttäjiä samanaikaisesti. Palvelinriippumattomuus oli kuitenkin eräs yrityksen komponentille asettamista tavoitteista, eikä se näin ollen tule täyttymään. Käyttöönoton epäonnistuessa myös itse komponentti jäi prototyyppi-asteelle, eikä sitä toistaiseksi toteutettu tuotantoon soveltuvaksi.

Tulevaisuudessa ongelma voi ratketa IPv6-osoiteavaruuden yleistyessä laajamittaiseen käyttöön, jolloin osoitteenmuunnos, jonka tehtävänä on varmistaa nykyisten IPv4-osoitteiden riittäminen, saattaa jäädä tarpeettomaksi ja voi mahdollisesti poistua käytöstä. Toisaalta on myös mahdollista, että nykyisin käytössä olevan Google Play Games SDK-komponentin markkinointistrategiaa muutetaan ja sen käyttäminen muuttuu maksulliseksi. Näissä esimerkinomaisissa tilanteissa Hyperkani Oy -yritys joutuu uudelleen arvioimaan vaihtoehtoisen komponentin käyttöönottoa. Tällöin reaaliaikainen moninpelirajapinta sisältää toimivan pohjan ongelman ratkaisemiseksi.

7. JÄRJESTELMÄN ARVIOINTIA

Tässä luvussa arvioidaan järjestelmän suorituskykyä kohdassa 4.4 sille asetettujen vaatimusten perusteella. Loppukäyttäjien eli pelaajien kannalta oleellisimmat seikat liittyvät luonnollisesti järjestelmän tiedonsiirtovaatimusten täyttymiseen, kun taas kehittäjien kannalta järjestelmän ylläpidettävyyteen vaikuttaa oleellisesti sen skaalautuvuus pelaajamäärien kasvaessa [34].

Muita järjestelmän arviointiin vaikuttavia osatekijöitä ovat esimerkiksi sen ylläpidettävyys, turvallisuus ja saavutettavuus. Nämä asiat ovat edellämainittujen ominaisuuksien lisäksi merkittävässä roolissa sekä kehittäjien että pelaajien kannalta, joten myös näitä tekijöitä on käsitelty lyhyesti tässä luvussa.

7.1 Käyttäjien väliset viiveet

Järjestelmää arvioitaessa on oleellista tietää miten järjestelmä suoriutuu käyttäjämäärien kasvaessa. Verkkopelaamisen kannalta järjestelmän suorituskyvyn mittareina voidaan käyttää pelaajien välille muodostuvia viiveitä [7]. Mittausten perusteella määritellään erilaisia järjestelmän käyttöön liittyviä suosituksia sekä rajoitteita esimerkiksi pakettien koon tai lähetysvälin suhteen.

Pelaajien kannalta viiveiden suuruus on eräs oleellisimmista moninpelijärjestelmän suorituskykyyn vaikuttavista ominaisuuksista, sillä kasvaessaan viiveet heikentävät pelikokemusta ja tekevät siitä pahimmillaan mahdotonta [2][5]. Järjestelmää suunniteltaessa viiveiden pysyminen maltillisina oli eräs tärkeimmistä vertaisverkkojen valintaperusteista, joten siksi on oleellista testata, miten käyttäjien välinen viive käyttäytyy pakettien koon kasvaessa tai pakettien välisen lähetysajan pienentyessä. Lisäksi näitä viiveitä vertaillaan asiakas-palvelin-arkkitehtuuria hyödyntävän relaypalvelimen kanssa suoritettuihin testeihin, jolloin saadaan todellinen kuva tämän ratkaisun soveltuvuudesta tarkoitukseensa.

Viiveiden suuruuteen vaikuttavat esimerkiksi käyttäjien välinen maantieteellinen sijainti, verkossa kulkevan muun liikenteen suuruus sekä välissä olevien tukiasemien, reitittimien tai muiden verkkolaitteiden määrä [7][29, s. 3-12]. Tästä johtuen kaikki

testit on suoritettu omassa suljetussa lähiverkossa, jolloin viiveet pysyvät pääosin pieninä ja ovat kuitenkin samalla vertailukelpoisia keskenään. Viiveiden mittaus on toteutettu ohjelmallisesti, joten saapuvien pakettien purkamiseen voidaan olettaa kuluvan hieman aikaa. Tämä ei kuitenkaan vaikuta tulosten vertailukelpoisuuteen. Vaikka mittaus toteutetaankin ainoastaan paikallisesti, sen perusteella havaitaan miten järjestelmä käyttäytyy kuormituksen kasvaessa.

Testauksen aikana viiveet mitattiin sekä vertaisverkkoyhteyksillä että relay-serverin kautta kulkevalla yhteydellä. Alkuperäisen Google Play -moninpelikomponentin suorituskyvyn mittaaminen koettiin lopulta tarpeettomaksi, sillä vertaisverkon suoran yhteyden muodostuksen onnistuessa laitteiden välisissä viiveissä ei havaittu järjestelmäkohtaisia eroja. Relay-serverin kautta kiertävät paketit puolestaan olivat Googlen tuotteella merkittävästi hitaampia, sillä kyseinen palvelin sijaitsee tracert-työkalun perusteella maantieteellisesti merkittävästi kauempana (Yhdysvalloissa), kuin tässä työssä toteutetun järjestelmän välityspalvelin.

Viiveiden lisäksi käyttäjän havaitsemaan pelikokemukseen vaikuttaa myös viiveisiin olennaisesti liittyvä suure *viiveen vaihtelu* (Jitter), jolla tarkoitetaan muutoksia pelaajan kokemien viiveiden suuruuksien välillä [5]. Jitterin kasvaessa yksittäiset paketit päätyvät vastaanottaville tahoille hyvinkin nopeasti, kun taas toisaalta joidenkin pakettien toimitus saattaa kestää useita sekunteja. Jitterin vaikutus pelikokemukseen on kuitenkin varsin riippuvainen siitä, miten kehittäjät ovat huomioineet viiveiden vaihtelun kehitystyössään [5]. Tästä johtuen jitterin suuruus ei sellaisenaan ole yhtä hyvä mittari järjestelmän suorituskyvyn mittaamiseksi kuin viiveet.

7.1.1 Pakettien koon vaikutus

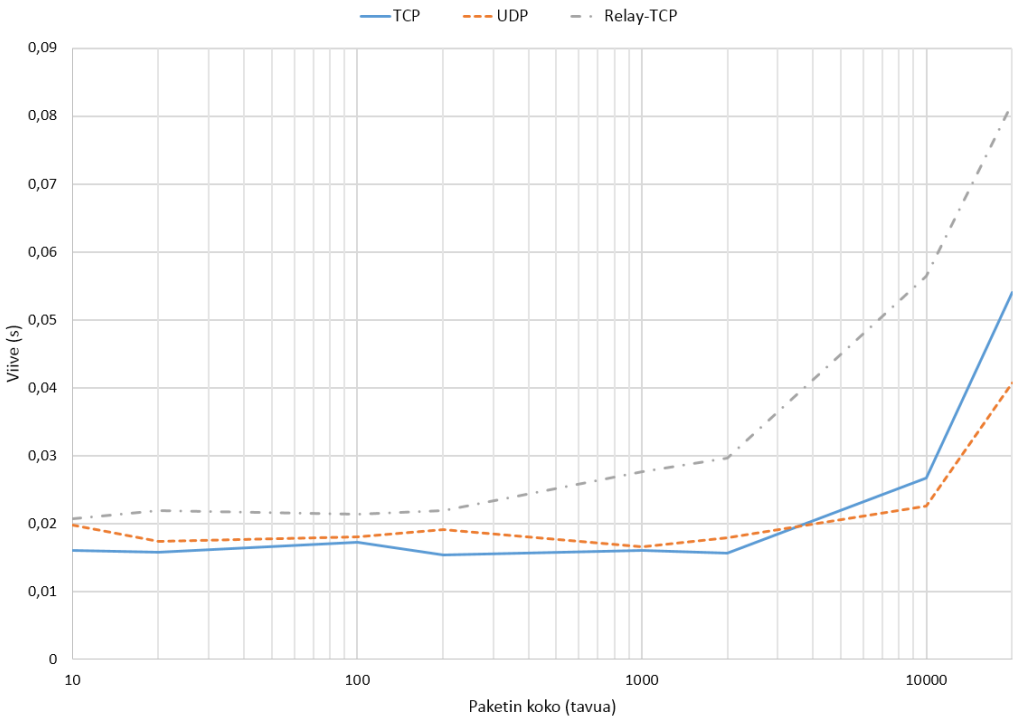
Ensimmäisessä testissä pidettiin pakettien välinen aika vakiona ja pakettien kokoa kasvatettiin ajan funktiona. Taulukossa 7.1 luetellaan tässä testissä käytetyt parametrit. Testin luotettavuuden lisäämiseksi testin jokainen askel suoritettiin useita kertoja ja lopputuloksena hyödynnettiin saatujen tulosten keskiarvoa.

Testin tulokset on esitetty kuvassa 7.1. Havaitaan, että pakettien koon pysyessä alle tuhannessa tavussa, viiveet pysyvät vakiona noin 20 millisekuntin suuruisina, kun taas pakettien koon kasvaessa yli tuhannen tavun myös viiveet alkavat nousemaan. Osasyy tähän voi olla 802.11-standardin mukaisen ethernet-kehyksen maksimikoko (1 518 tavua) [15], jolloin tätä suuremmat paketit joudutaan hajauttamaan useampaan osaan. Vastaavasti myös erillisen relay-palvelimen kautta kierrätettävät paketit kasvattavat viiveen suuruutta, sillä nämä paketit käsitellään kertaalleen ensin relay-

palvelimella, ennen kuin ne voidaan edelleenlähettää vastaanottavalle osapuolelle.

Taulukko 7.1: *Parametrit pakettien koon vaikutuksen testaamiseen*

Pakettien välinen lähetysväli	**200ms**
Käytetyt protokollat	**TCP, UDP, TCP relay-palvelimen kautta**
Paketin sisältöjen suuruudet tavuina	**10, 20, 100, 200, 1 000, 2 000, 10 000, 20 000**
Lähetettävien pakettien määrä jokaista testiä kohden	**200**



Kuva 7.1: *Viiveiden käyttäytyminen paketin koon mukaan*

Testin perusteella voidaan todeta järjestelmän toimivan tehokkaimmin silloin, kuin pakettien koot eivät ylitä ethernet-kehyksen maksimikokoa. Pakettien kokojen kasvaessa yli ethernet-kehyksen maksimikoon viiveet alkavat nousemaan. Käyttöjärjestelmä voi hajauttaa myös pienemmät tietomäärät useampaan pakettiin [59], joka osaltaan vaikuttaa viiveiden kasvamiseen. Mielenkiintoista on kuitenkin havaita, että myös Google Play rajoittaa oman moninpelikomponenttinsa pakettien koot TCP-protokollaa hyödynnettäessä 1400 tavuun ja UDP-protokollalla 1168 tavuun [24].

7.1.2 Pakettien tiheyden vaikutus

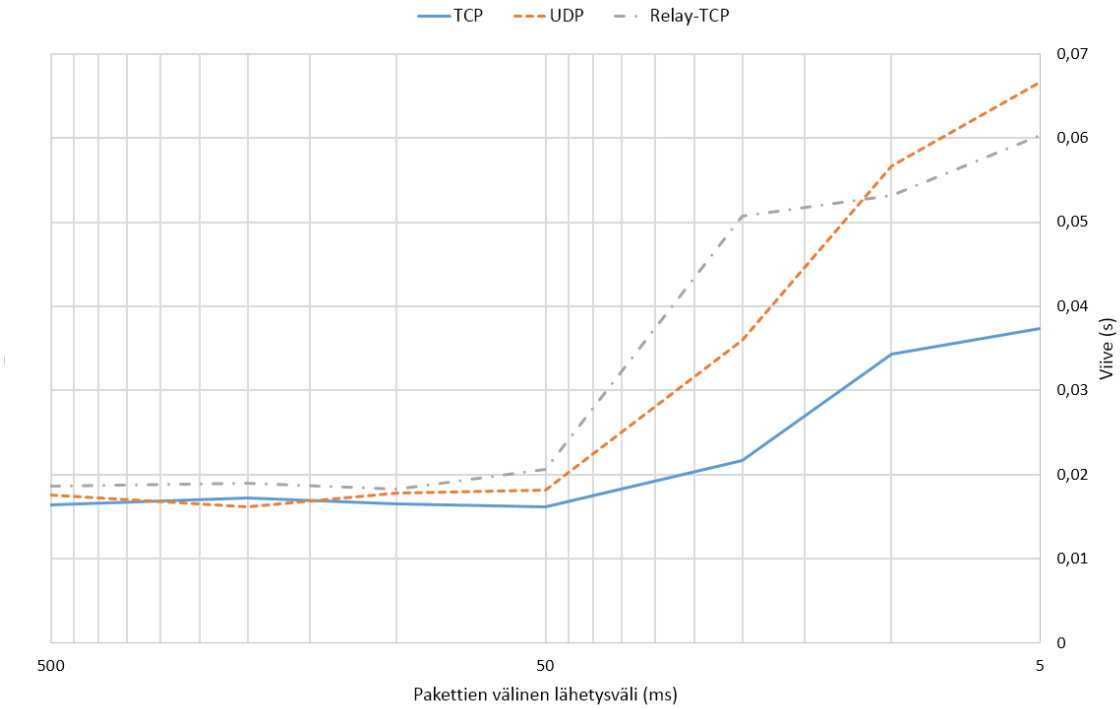
Toisessa testissä simuloidaan tilannetta, jossa pakettien koko pysyy vakiona, mutta pakettien lähetysväliä pienennetään ajan funktiona. Pakettien tiheä lähetysväli on nopeatempoisissa verkkopeleissä varsin tavanomaista, joten on oleellista tietää miten järjestelmä käyttäytyy lähetysvälin ollessa erittäin pieni. Taulukossa 7.2 esitetään tämän testin parametrit. Edellisen testin tavoin testin jokainen askel suoritetaan useita kertoja luotettavamman tuloksen aikaansaamiseksi.

Taulukko 7.2: *Parametrit pakettien tiheyden vaikutuksen testaamiseen*

Pakettien koko	**100 tavua**
Käytetyt protokollat	**TCP, UDP, TCP relay-palvelimen kautta**
Eri lähetysvälit	**500ms, 200ms, 100ms, 50ms, 20ms, 10ms, 5ms**
Lähetettävien pakettien määrä jokaista testiä kohden	**200**

Testien tulokset on esitetty kuvassa 7.2. Havaitaan, että pakettien lähetysvälin pienentyessä alle 50 millisekunnin viiveet alkavat kasvamaan lineaarisesti. Osasyynä

tähän voi olla käyttöjärjestelmän toiminta useita perättäisiä paketteja lähetettäessä, jolloin näitä paketteja yhdistellään mielivaltaisesti toimintojen nopeuttamiseksi. Toinen, kenties realistisempi vaihtoehto on yksinkertaisesti verkon kuormittuminen yli sen maksimikapasiteetin, joka näkyy pelaajille kasvaneina viiveinä.



Kuva 7.2: *Viiveiden käyttäytyminen pakettien tiheyden vaikutuksesta*

Testistä voidaan todeta järjestelmän viiveiden pysyvän lähes vakiona pakettien välisen lähetysvälin ollessa kontrolloituna yli 50 millisekunnin. Pienemmät lähetysvälit aikaansaavat suurempia viiveitä. Lisäksi pakettien välisen lähetysvälin pienentyessä alle 50 millisekunnin paketteja alkoi hukkumaan. Tämä selittyisi sillä, että saapuvat paketit jäävät mahdollisesti käyttöjärjestelmän verkkototeutuksen puskuriin odottelemaan käsittelyä, kunnes puskurin maksimikoon ylittyessä saapuvia paketteja aletaan hylkäämään ja *packet loss* kasvaa.

7.2 Skaalautuvuus käyttäjämäärien kasvaessa

Vertaisverkkoihin perustuvan moninpelijärjestelmän selkeä etu asiakas-palvelin-arkkitehtuuriin verrattuna on sen skaalautuvuus käyttäjämäärien kasvaessa. Vertaisverkkojen ansiosta käytössä olevalle matchmaking-palvelimelle muodostetut yhteydet eivät ole pitkäkestoisia eikä ruuhkaa näinollen pääse syntymään, kun taas taval-

lisen pelipalvelimen tapauksessa järjestelmän skaalautuvuus on suoraan verrannollinen käytössä olevien palvelinten määrään. [34]

Mikäli kahdelle eri pelipalvelimelle yhdistäneet pelaajat haluaisivat pelata yhdessä, nämä erilliset pelipalvelimet tulisi myös saattaa toimimaan keskenään, josta seuraa lisää haasteita pelitilojen väliseen synkronointiin. Lisäksi asiakasohjelmien tulisi olla tietoisia muiden käytössä olevien palvelinten yhteysmääristä ruuhkien välttämiseksi palvelinta valittaessa. Vertaisverkkoja hyödynnettäessä näitä ongelmia ei ole.

Vaikka skaalautuvuus ei sellaisenaan olekaan mitattava suure, voidaan vertaisverkkoihin perustuvaa moninpelijärjestelmää edellämainituista syistä johtuen pitää yleisessä tapauksessa skaalautuvampana kuin sellaista järjestelmää, jonka toimintaperiaate perustuu asiakas-palvelin-arkkitehtuuriin.

7.3 Ylläpidettävyys, tietoturva ja saavutettavuus

Eri arkkitehtuuriin perustuvia järjestelmiä vertailtaessa on oleellista huomioida myös muita osa-alueita, joita ei voida sellaisenaan mitata. Hyperkani Oy -yrityksen puolesta eräitä oleellisimpia osa-alueita ovat tällöin ylläpidettävyys, tietoturva sekä saavutettavuus.

Järjestelmän ylläpidettävyys helpottuu vertaisverkkoihin perustuvan järjestelmän tapauksessa selkeästi, sillä ylläpidettävänä on ainoastaan yksi käytössä oleva matchmaking palvelin. Useamman palvelimen tapauksessa esimerkiksi palvelinohjelmistoon tai palvelinjärjestelmiin kohdistuvat muutokset tulee suorittaa jokaiselle käytössä olevalle järjestelmälle erikseen. Useamman palvelimen hankinta- sekä jatkuvat ylläpitokustannukset ovat myös huomattavasti korkeammat kuin yksittäisen palvelimen tapauksessa.

Järjestelmän tietoturvaa tarkasteltaessa vertaisverkkoja hyödyntävät pelit vaativat kehittäjiltään enemmän erilaisia tietojen oikeellisuuksien tarkasteluja asiakas-palvelin-arkkitehtuuriin verrattuna, sillä pelien tilatietojen hallinta on jaettu vertaisverkon eri tahojen kesken. Tästä johtuen mahdolliset hyökkääjät voivat esimerkiksi pyrkiä käyttämään muokattua asiakasohjelmaa pelin pelaamiseen. Tällä pyritään saavuttamaan jonkinlaista etua muihin pelaajiin nähden. Jakamalla pelien tilatiedot kaikkien vertaisverkon tahojen kesken, voidaan yksittäiset huijausyritykset tunnistaa ja eliminoida. Säilytettäessä tilatietoja erillisellä pelipalvelimella samaista ongelmaa ei sellaisenaan esiinny, mutta julkinen palvelin on puolestaan alttiina muille hyökkäyksille. [33]

Saavutettavuuden kannalta yksittäinen matchmaking-palvelin on kuitenkin altis erilaisille palvelunestohyökkäyksille tai esimerkiksi sähkökatkoille, sillä palvelimen ollessa saavuttamattomissa ei peliseuran löytäminen eikä pelien pelaaminen onnistu. Tästä johtuen voidaankin pitää oleellisena useamman matchmaking-palvelimen pystyttämistä saavutettavuuden kasvattamiseksi, joka on edelleen resurssien kannalta edullisempaa kuin vielä useamman erillisen pelipalvelimen ylläpito. Yksittäinen matchmaking-palvelin ei sellaisenaan takaa järjestelmän saavutettavuutta.

7.4 Johtopäätökset

Viivetestien perusteella havaitaan, että järjestelmä täyttää sille asetetut tiedonsiirtoa koskevat vaatimukset. Testien tuloksista voidaan myös päätellä, että vertaisverkot mahdollistavat nopeamman tiedonsiirron sekä tasaisemman kuormituksen pelaajien välillä asiakas-palvelin-arkkitehtuuriin verrattuna.

Järjestelmää käyttöönotettaessa matchmaking-palvelimia vaaditaan saavutettavuuden kannalta useampia, mutta palvelinten alhaisten suorituskyky- sekä yhteysvaatimusten johdosta palvelimet eivät vaadi juuri ollenkaan yrityksen resursseja. Lisäksi se on ylläpitäjien näkökulmasta skaalautuva ja helposti ylläpidettävä pelaajamäärien kasvaessa. Tietoturvasta huolehtiminen jää sellaisenaan kehittäjien vastuulle. Suoritettujen testien sekä muun arvioinnin perusteella voidaan reaaliaikaisen moninpelirajapinnan prototyyppiä pitää tarkoitukseensa soveltuvana.

8. JÄRJESTELMÄN KEHITYSKOHTEET

Reaaliaikaisen moninpelirajapinnan prototyyppiä ei yhteydenmuodostuksen ongelmallisuudesta johtuen toteutettu tuotantokäyttöön soveltuvaksi. Mikäli rajapinnasta halutaan toteuttaa lopullinen tuotantoon soveltuva versio, tulee kehittäjien pohtia ainakin seuraavia osa-alueita.

8.1 Yhteyden muodostamisen kiertotiet

Suorien yhteyksien muodostumisen epäonnistuminen on selkeästi merkittävin tekijä sille, ettei Hyperkani Oy ottanut järjestelmää toistaiseksi käyttöön. Tästä johtuen tässä luvussa tutkitaan erilaisia yhteyden muodostumiseen soveltuvia menetelmiä sekä pohditaan niiden hyödyntämistä käytännössä.

Kohdassa 4.2 käsitellyn verkon analysointityökalulla tuotetun datan perusteella havaitaan, että myös Google Play Games Services SDK:lla toteutettu mobiilimoninpeli Bomber Friends joutuu jokaista pelaajaparia varten muodostamaan TCP-yhteyden erillisten palvelinten kautta suoran yhteydenmuodostuksen epäonnistuessa. Mielenkiintoista järjestelmän toimintaperiaatteissa on kuitenkin se, että toisin kuin Libniceä hyödyntävällä prototyypillä, UDP-yhteys saatiin usein muodostettua suoraan parien välille ilman tarvetta kierrättää paketteja erillisen välityspalvelimen kautta, vaikka molemmat laitteet sijaitsisivatkin osoitteenmuunnosten takana.

8.1.1 Eri tyyppiset osoitteenmuunnokset

Tässä luvussa läpikäytävät menetelmät, joiden avulla suora yhteys kahden osoitteenmuunnosten takana sijaistevien tahojen välille pyritään muodostamaan, ovat riippuvaisia käytettävien osoitteenmuunnosten toimintaperiaatteista [50]. Nämä toimintaperiaatteet voidaan lähteiden [50] ja [57] mukaan luokitella neljään eri kategoriaan:

- *Full Cone*: Tätä menetelmää noudattava osoitteenmuunnin laatii kiinteät reititystaulut aliverkosta löytyvän tahon A ja osoitteenmuunnoksen ulkoisten

osoite-portti-parien välille. Esimerkiksi tahon A lähettäessä portista 8000 osoitteenmuunnoksen N (202.123.211.25) läpi dataa, se voidaan reitittää esimerkiksi portin 12345 kautta. Tällöin kuka tahansa ulkopuolinen taho voi lähettää suoraan taholle A dataa lähettämällä paketin tähän tunnettun osoitteenmuunnoksen N avattuun osoite-portti-pariin (202.123.211.25:12345) lähdeosoitetiedoista riippumatta.

- *Restricted Cone*: Tämä hieman rajoittavampi menetelmä ei päästä tuntemattomasta IP-osoitteesta saapuvia paketteja aiemmin laaditun reitityksen läpi, ellei kyseisen reitityksen omaava aliverkon taho A ole lähettänyt myös tälle kohteelle dataa samasta sisäisestä osoite-portti -parista. Esimerkiksi tahon A lähettäessä portista 8000 osoitteenmuunnoksen N (202.123.211.25) läpi dataa taholle B osoitteenmuunnoksen reitittämän portin 12345 kautta, ei uusi taho C kuitenkaan onnistu lähettämään taholle A porttiin 8000 paketteja osoitteeseen 202.123.211.25:12345, ennenkuin taho A on lähettänyt myös taholle C ainakin yhden paketin.

- *Port Restricted Cone*: Tämä edellisen kanssa lähes identtinen mutta vielä rajoittavampi menetelmä tarkastelee aiemmin lähetettyjen kohdeosoitteiden lisäksi myös lähetettyjen pakettien kohdeportteja, eikä päästä tunnetustakaan kohdeosoitteesta paketteja reitityksen läpi, mikäli ne tulevat ennalta tuntemattomasta portista. Toisin sanoen tahon A lähettäessä osoitteenmuunnoksen N (202.123.211.25) läpi dataa tahon B porttiin 80, ei tahon B portista 90 lähettämiä paketteja osoitteeseen 202.123.211.25:12345 reititetä taholle A niiden tuntemattoman lähdeportin takia.

- *Symmetric*: Symmetrinen NAT eroaa aikaisemmista menetelmistä siten, että jokaista aliverkosta ulkoverkkoon muodostettavaa yhteyttä varten laaditaan uusi reitityssääntö, joka on käytössä ainoastaan näiden kahden tahon välisessä kommunikoinnissa. Muista osoitteista tai porteista saapuvat yhteydet estetään. Tämä tarkoittaa sitä, että tahon A muodostaessa yhteyden osoitteenmuunnoksen N läpi tahoon B, sille voidaan avata osoitteenmuunnoksesta esimerkiksi portti 12345. Tahon A lähettäessä samasta lähdeportista uudelle taholle C paketteja, uusi reitityssääntö laaditaan esimerkiksi portin 23456 kautta. Vaikka nämä molemmat reitityssäännöt ovatkin yhtäaikaisesti voimassa, ei taho B voi vastata taholle A osoitteenmuunnoksen N portin 23456 kautta.

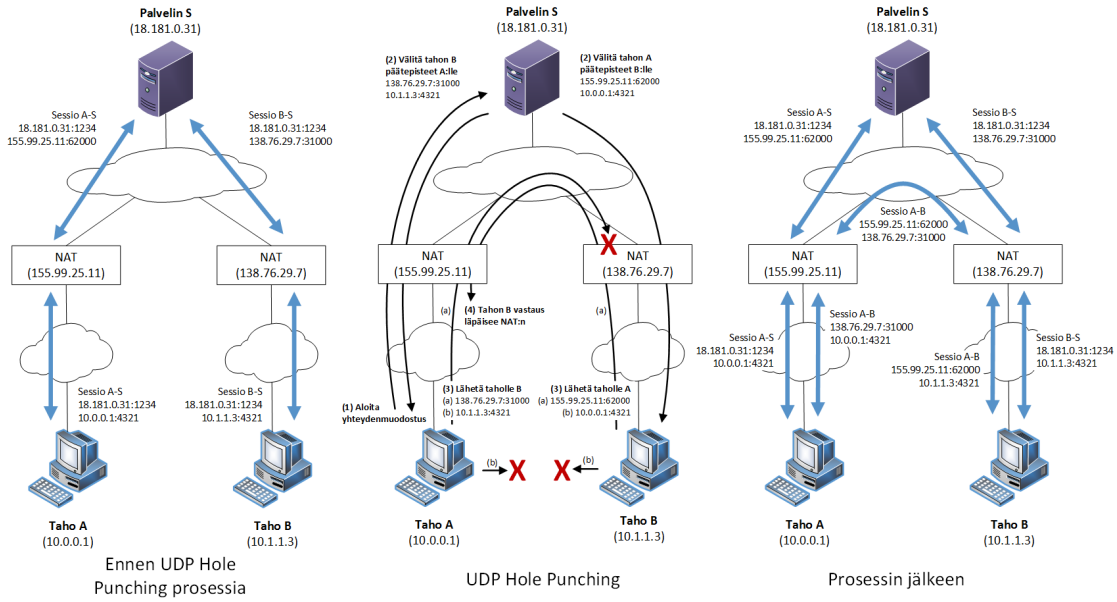
Näistä Full Cone on sallivin ja Symmetric rajoittavin menetelmä. Tavallisesti osoitteenmuunnoksen reititys aukeaa, kun aliverkon sisällä oleva taho lähettää ensimmäisen paketin ulkoverkkoon kyseisen osoitteenmuunnoksen läpi [57]. Tästä johtuen

reititykseen liittyy tavallisesti aikakatkaisu, joka purkaa reitityksen, mikäli dataa ei liiku kyseisen reitityksen läpi.

Vaihtoehtoisesti osoitteenmuunnoksen asetuksista voidaan tavallisesti kytkeä päälle myös staattinen reititys (port forwarding), jossa reititys pidetään jatkuvasti auki myös ulkoverkosta tulevia yhteyden avauksia varten [57]. Tällöin nämä kyseiseen osoitteenmuunnoksen osoite-portti -pariin saapuvat paketit ohjataan suoraan reitityksen määräämälle aliverkon taholle lähettäjästä riippumatta. Staattisen reitityksen käyttäminen ei kuitenkaan ole tavallisimmissa kulutajille suunnatuissa kytkimissä ja reitittimissä tehdasasetuksilla käytössä, mikä tekee sen käyttämisestä varsin harvinaista.

8.1.2 UDP hole punching

UDP hole punching -tekniikalla kaksi osoitteenmuunnosten takana olevaa tahoa saattavat onnistua muodostamaan keskenään suoran UDP-yhteyden välissä olevien osoitteenmuunnoksien toimintaperiaatteista riippuen. Tekniikka perustuu molempien tahojen tunteman julkisen yhteydenmuodostusta avittavan palvelimen toimintaan, UDP-protokollan tilattomaan luonteeseen sekä osoitteenmuunnosten suorittavien laitteiden toimintaperiaatteiden ennalta-arvattavuuteen. Menetelmä käydään läpi lähteen [17] mukaisesti.



Kuva 8.1: *UDP hole punching -toimintaperiaate [17]*

Yleisimmässä tapauksessa yhteydenmuodostusta yrittävät tahot A ja B ovat yksittäisten osoitteenmuunnosten takana kuvan 8.1 ensimmäisen vaiheen mukaisesti. Koska palvelin S omaa julkisen IP-osoitteen, molemmat tahot voivat muodostaa siihen UDP-yhteyden, jolloin välissä olevat osoitteenmuuntajat avaavat jonkun tietyn portin toimintaperiaatteensa mukaisesti näitä yhteyksiä varten. Tässä esimerkissä palvelimen S tulee vastata tahon A viesteihin osoitteeseen 155.99.25.11 porttiin 62000 ja tahon B kyselyihin osoitteeseen 138.76.29.7 porttiin 31000. Osoitteenmuunnokset ohjaavat jatkossa näihin omiin julkisiin osoite- ja porttipareihin saapuvat paketit oikeille tahoilleen, kunhan ne tulevat sellaisesta IP-osoitteesta, johon taho on jo aiemmin luonut yhteyden.

Palvelimen havaitessa saapuvat UDP-paketit, se lukee viestien otsikkotiedoista molempien tahojen julkiset pääteosoite- ja porttiparit vastauslähetyksiä varten. Lisäksi viestien sisältöihin on liitetty molempien tahojen yksityiset osoitetiedot. Molempien tahojen osoitetiedot saatuaan palvelin S välittää nämä tiedot tahoille ristiin siten, että tahon A vastaukseen liitetään tahon B julkinen sekä yksityinen osoite-porttipari sekä vastaavasti taholle B tahon A molemmat osoitetiedot.

Olettaen tahon A saavan vastauksensa ensin, se pyrkii muodostamaan UDP-yhteyden tahon B julkisen- sekä yksityisten osoitetietojen perusteella kuvan 8.1 vaiheen 2 mukaisesti. Koska tahot eivät sijaitse samassa aliverkossa, yksityisten osoitetietojen perusteella lähetetty paketti ei koskaan saavuta oikeaa kohdekonetta. Julkiseen osoitteeseen kohdistettu paketti sen sijaan kulkee tahon A osoitteenmuunnoksen läpi avaamalla portin saapuville yhteyksille tahon B julkisesta IP-osoitteesta 138.76.29.7. Paketti pysähtyy kuitenkin tahon B osoitteenmuunnokseen sen saapuessa laitteeseen tuntemattomasta IP-osoitteesta.

Tahon B saadessa vastauksensa palvelimelta S se lähettää tahon A tavoin sekä tahon A julkiseen että yksityiseen osoitteeseen vastaavanlaisen UDP-paketin. Yksityiseen osoitteeseen lähetetty paketti ei luonnollisesti jälleen koskaan saavu toivottuun kohdeosoitteeseen. Tahon A julkiseen osoitteeseen lähetetty paketti sen sijaan kulkee ensin tahon B osoitteenmuunnoksen läpi, jossa sen otsikkotietoihin vaihdetaan tahon B julkinen osoite, jonka jälkeen se saapuu tahon A osoitteenmuunnokselle. Koska taho A on jo aiemmin lähettänyt tahon B julkiseen osoitteeseen viestin, tulkitaan nyt saapuva UDP-paketti vastaukseksi tähän viestiin ja se ohjataan toivotusti osoitteenmuunnoksen läpi taholle A. Tämän jälkeen myös tahon A viestit kulkevat vastaavasti tahon B osoitteenmuunnoksen läpi kuvan 8.1 vaiheen 3 mukaisesti.

Menetelmän onnistuminen riippuu käytettyjen osoitteenmuunnosten toimintaperi-

aatteista. Mikäli osoitteenmuunnos toimii kohdassa 8.1.1 kuvatun Full Cone-, Restricted Cone tai Port Restricted Cone -tekniikoiden toimintaperiaatteiden mukaisesti, menetelmä toimii sellaisenaan. Symmetrinen osoitteenmuunnos sen sijaan tuottaa ongelmia. Tällöin tahojen toisilleen lähettämät viestit eivät ikinä päädy perille, sillä näitä yhteyksiä varten on avattu laitteeseen uudet tuntemattomat portit, joita nämä tahot eivät luonnollisestikaan tiedä. [17]

Useat tällaiset symmetriset osoitteenmuuntajat, jotka reitittävät eri kohteeseen lähetetyt yhteydet aina eri portin kautta, toimivat kuitenkin suhteellisen ennaltaarvattavasti. Aiemmin kuvatussa esimerkissä tahon A avatessa yhteyden tahon B osoitteenmuuntajaan se oltaisiin esimerkiksi voitu reitittää portin 62001 kautta aiemmin käytetyn portin 62000 sijaan. Tällä tai vastaavalla pääteltävissä olevalla logiikalla toimivat osoitteenmuuntajat on myös mahdollista saattaa toimimaan, mutta menetelmä on epävarmempi ja sisältää useampia tilanteeseen vaikuttavia muuttujia. On esimerkiksi mahdollista, että tämä seuraava ennustettava porttinumero ehditäänkin reitittämään jo johonkin toiseen aliverkon osoitteeseen odotettaessa vastausta palvelimelta S. Lisäksi tarvitaan ainakin kaksi tai jopa useampia eri osoitteissa sijaitsevia palvelimia uusien UDP-yhteyksien avaamiseksi, jotta uusien porttinumeroiden seuraama logiikka voidaan päätellä. [17]

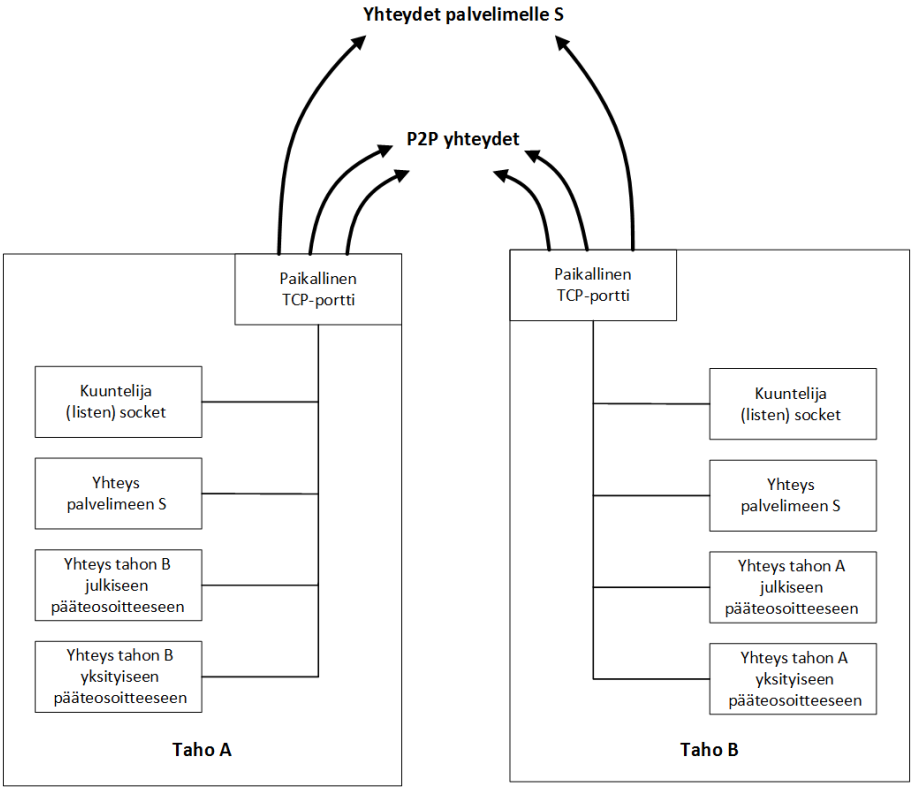
UDP hole punching on kuitenkin käyttökelpoinen menetelmä tahojen välisten UDP-yhteyksien luomiseen tilanteissa, joissa molemmat tahot ovat tällaisten vertaisverkkojen muodostamista tukevien osoitteenmuunnosten takana. Menetelmän yksinkertaisuudesta sekä suuresta onnistumisprosentista [17] johtuen onkin syytä uskoa, että myös Google Play Games Services SDK hyödyntää tätä menetelmää UDP-yhteyksien luomiseen. Verkon analysointityökalun avulla tämä olisi myös mahdollista selvittää seuraamalla yhteyden avauksessa lähetettyjen pakettien kohdeosoitteita sekä sisältöjä. Mikäli Hyperkani Oy -yritys ottaa reaaliaikaisen moninpelirajapinnan käyttöön, tämä menetelmä tulisi toteuttaa yhteyden muodostamista varten.

8.1.3 TCP hole punching

TCP hole punching -tekniikan perusperiaate on protokollatasolla hyvin samankaltainen kuin UDP-yhteydenkin kanssa, mutta selkeästi pienempi osa osoitteenmuunnoslaitteista tukee tätä tekniikkaa ja siksi sen onnistuminen on epätodennäköisempää. Lisähaasteita menetelmän käyttöön tuo se, että standardoitu Berkley sockets API-rajapinta on suunniteltu asiakas-palvelin-arkkitehtuurin pohjalta, eikä se sellaisenaan mahdollista yhteyden avaamista eri kohteisiin sekä yhteyksien kuuntelua samasta yksittäisestä portista. Suurin osa käyttöjärjestelmistä kuitenkin mahdollistaa

tämän SO_REUSEADDR-vipua käyttämällä TCP-socketien yhteydessä. Menetelmä käydään läpi lähteen [17] mukaisesti.

Menetelmässä hyödynnetään jälleen erillistä julkista palvelinta S julkisten sekä yksityisten osoitetietojen vaihtamiseen samoin kuin edellämainitussakin tekniikassa. Osoitetiedot vaihdettuaan tahot yrittävät muodostaa yhteyttä samoista aiemmin palvelimen S kanssa kommunikointiin käytetyistä paikallisista TCP-porteista toistensa sekä julkisiin että yksityisiin IP-osoitteisiin. Lisäksi tahot joutuvat kuuntelemaan samoja portteja saapuvia yhteyksiä varten. Kuva 8.2 havainnollistaa tahojen samanaikaisesti auki olevia yksittäiseen porttiin sidottuja yhteyksiä.



Kuva 8.2: *TCP hole punching -toimintaperiaate [17]*

Yhteyden muodostaminen noudattaa samoja periaatteita kuin UDP-yhteyksilläkin: ensimmäisenä vastauksen saanut taho, oletuksena taho A, yrittää muodostaa TCP-virran lähettämällä SYN-paketin tahon B julkiseen osoitteeseen. Paketti kuitenkin pysähtyy tahon B osoitteenmuunnokseen sen saapuessa laitteelle tuntemattomasta osoitteesta. Tahon B yhteydenmuodostus tahoon A kuitenkin onnistuu tämän jälkeen, sillä tahon A osoitteenmuunnokseen on edellisen lähetetyn paketin perusteella luotu sääntö tahon B julkisesta osoitteesta saapuville paketeille, mikäli osoitteenmuunnos ei ole symmetrinen.

Yhteyttä yrittävät ohjelmat havaitsevat asian jommalla kummalla seuraavista tavoista:

- Tahon A TCP-toteutus havaitsee taholta B vastaanotetun SYN-paketin ja tulkitsee sen vastaukseksi aiemmin lähetetylle SYN-paketille. Tämän ansiosta tahon A connect-funktiokutsu tahon B julkiseen osoitteeseen onnistuu ja tahon A kuuntelija-socket jää edelleen odottamaan. Lisäksi taho A vastaa tahon B SYN-pakettiin SYN-ACK-paketilla, jonka saapuessa myös taho B havaitsee yhteydenmuodostumisen onnistuneen.
- Tahon A kuuntelija-socket voi havaita taholta B saapuvan SYN-paketin, jolloin tahon A accept-funktiokutsu palauttaa tämän uudeksi tulkitun yhteyden. Lisäksi taho A vastaa viestiin SYN-ACK-paketilla, jolloin myös taho B havaitsee yhteydenmuodostumisen onnistuneen. Tämä menetelmä muistuttaa tavallisen asiakas-palvelin-mallin yhteydenmuodostusta.

Lisäksi on myös mahdollista, että molempien tahojen lähettäessä ensimmäistä SYN-pakettiaan, ne ajoittuvat ajallisesti riittävän samanaikaisesti siten, että molempien tahojen lähettämät paketit ehtivät kulkemaan tahoja vastaavien omien osoitteenmuunnosten läpi ennen kuin kummankaan lähettämä paketti saapuu vastapuolen osoitteenmuunnokseen. Yhteyttä yrittävät ohjelmat havaitsevat tilanteen jälleen jommalla kummalla edellämainituista tavoista ja tästä johtuen onkin mahdollista, että järjestelmien TCP-toteutuksista riippuen molempien järjestelmien connect-kutsut saattavat epäonnistua ja accept-kutsut onnistua. Lopputuloksena on joka tapauksessa toimiva TCP-virta tahojen välillä.

Onnistuessaan TCP hole punching -menetelmän avulla muodostettu yhteys saattaa olla jopa vakaampi kuin UDP:lla muodostettu vastaava, sillä TCP:n tilallisesta luonteesta johtuen molemmat osoitteenmuuntajat osaavat päätellä yhteyden keston ja purkaa muunnoksen tämän jälkeen hallitusti. Tämä menetelmä on kuitenkin vaikeampi toteuttaa ja harvemmat osoitteenmuuntajat tukevat sen toimintaa. Myöskään Google Play Games Services ei onnistunut kertaakaan luomaan suoraa TCP-yhteyttä kahden tahon välille. Testien mukaan se toimii kuitenkin 64% tapauksista [17]. Pelin luonteesta riippuen suoran TCP-yhteyden luominen tahojen välille voi myös olla tarpeetonta, sillä reaaliaikakriittisen sisällön jakamiseen suora UDP-yhteys suoriutuu tehtävästään paremmin, kun taas tärkeämmät TCP-viestit voidaan tarvittaessa kierrättää erillisen relay-palvelimen kautta.

8.1.4 Muut menetelmät

Edellämainittujen tekniikoiden lisäksi on olemassa myös muita tapoja vertaisverkkojen tahojen välisten suorien yhteyksien muodostamiseen. Näistä yksinkertaisin kulkee nimellä Connection Reversal. Menetelmän toimintaperiaate perustuu lähde- ja kohdetahon roolien vaihtamiseen tilanteissa, jossa yhteyden avaava taho ei ole joko ollenkaan tai on mahdollisimman sallivan osoitteenmuunnoksen takana [47]. Mikäli lähdetahon kohdetahoon avaama yhteys pysähtyy kohdetahon osoitteenmuunnokseen, voi kohdetaho kuitenkin joissakin tapauksissa muodostaa yhteyden lähdetahoon. Menetelmää voidaan tehostaa esimerkiksi STUN-protokollalla, jolloin ennen ensimmäistä yhteysyritystä selvitetään etukäteen, onko tahon ja julkisen internetin välissä osoitteenmuunnosta vai ei.

Toinen menetelmä sisältyy UPnP-protokollaperheen (Universal Plug and Play) perustoiminnallisuuksiin, jossa protokollaa tukevat reitittimet ja kytkimet sallivat paikallisten verkon tahojen kommunikoida osoitteenmuuntajan kanssa sekä esimerkiksi muokata tämän asetuksia [57]. Menetelmän avulla tahot voivat muun muassa avata uusia osoite- ja porttipareja osoitteenmuuntajalle sekä reitittää näihin saapuvan liikenteen itselleen, jolloin yhteyksien muodostaminen ulkoverkosta on mahdollista. Vaikka menetelmä onkin yleistymässä, käytössä on kuitenkin suuri määrä sellaisia osoitteenmuunnoksia, jotka eivät tue tätä protokollaa. Lisäksi menetelmä ei toimi ketjutettujen osoitteenmuunnosten yhteydessä, joten sen käyttökelpoisuus on toistaiseksi varsin marginaalista.

8.2 Tuki useammalle matchmaking-palvelimelle

Järjestelmän saavutettavuuden ollessa toistaiseksi ainoastaan yksittäisen matchmaking-palvelimen varassa, useamman palvelimen käyttöönotto on tuotantoon siirryttäessä pakollista. Useamman yhtäaikaisen palvelimen ollessa käytössä tulee miettiä, ovatko muut tarjolla olevat palvelimet ainoastaan varasijoilla, vai tasataanko pelikaverien etsimisestä syntyvää kuormaa hajauttamalla pelien etsijät useamman palvelimen kesken. Mikäli pelaajat hajautetaan eri palvelimille, tulee palvelimet saada jotenkin kommunikoimaan keskenään, tai vaihtoehtoisesti tiettyyn palvelimeen yhdistäneet pelaajat löytävät ainoastaan muita samaan palvelimeen yhdistäneitä pelaajia. Tämä voi toisaalta olla myös tarkoituksenmukaista palvelimien ollessa esimerkiksi maantieteellisesti sijoiteltuna, jolloin varmistetaan, että pelikaverit löytyvät samalta alueelta viiveiden minimoimiseksi.

8.3 Tuki useammalle Relay-palvelimelle

Vertaisverkkoyhteyksien luonnin epäonnistuessa pelaajien välinen tietoliikenne kierrätetään erillisen julkisesti saatavilla olevan relay-palvelimen kautta, johon molemmat pelaajat yhdistävät tiedonsiirtoa varten. Matchmaking-palvelinten tavoin tällaisia palvelimia tarvitaan käyttäjämäärien kasvaessa useampia. Lisäksi relay-palvelinta valittaessa on oleellista, että palvelin on maantieteellisesti mahdollisimman optimaalisesti sijoiteltuna molempien pelaajien kannalta viiveiden hallitsemiseksi.

Relay-palvelimen valinnan ollessa matchmaking-palvelinten valintaa kriittisempää, pelaajien välille tarvitaan jonkinlainen logiikka optimaalisen palvelimen valitsemiseksi. Tämä voidaan toteuttaa vaihtoehtoisesti joko heti matchmaking-palvelimella, jolloin pelaajien tietoja vaihdettaessa pelaajille ilmoitetaan jo tarvittavat relay-palvelimet kunkin tahon suhteen, tai erillisellä master-palvelimella, joka päättää kunkin pelaajaparin suhteen kulloinkin käytettävän relay-palvelimen.

8.4 Tuki useamman pelaajan väliselle vertaisverkolle

Tällä hetkellä testauksessa oleva prototyyppi mahdollistaa vertaisverkon luonnin ainoastaan kahden pelaajan välille. Tämä ratkaisu on toimiva kahden pelaajan välisten verkkopelien suhteen, mutta käytännössä järjestelmän tulee tukea myös ainakin neljän ja mielellään jopa kahdeksan pelaajan välisiä vertaisverkkoja. Tämän ominaisuuden toteuttaminen on kuitenkin yksinkertaista ja vaatii ainoastaan useamman NiceAgent-olion luomista moninpelirajapintaa alustettaessa.

8.5 Tuki huoneisiin liittymiselle kesken pelin

Yrityksen toiveena oli, että olemassa oleviin pelihuoneisiin voisi liittyä kesken jo käynnissä olevan pelin. Tämä on toistaiseksi mahdotonta Google Play Multiplayer SDK-rajapinnalla. Ominaisuuden toteuttaminen vaatii vertaisverkkojen ylläpitäjiksi (master) valittuja tahoja yhdistämään tarvittaessa uudelleen matchmaking-palvelimelle kesken käynnissä olevan pelin uusien pelaajien etsimistä varten.

Yksittäisen tahon lopettaessa pelisessionsa ja katkaisemalla yhteytensä olemassa olevan pelin aikana, vertaisverkon tahojen keskuudesta päätetty verkon ylläpitäjä tiedottaa tästä matchmaking-palvelimelle ilmoittamalla vapaana olevien vertaisverkon tahojen määrän. Matchmaking-palvelin hyödyntää tätä tietoa pelejä etsiessään ja sopivan kandidaatin löydyttyä ilmoittaa edellämainitulle vertaisverkon ylläpitäjälle

uuden pelaajan tiedot. Vertaisverkon ylläpitäjä sekä uusi pelaaja muodostavat yhteyden ensin keskenään ja vaihtavat tämän jälkeen muut peliin liittyvät tiedot, kuten esimerkiksi pelin tämän hetkisen tilan sekä muiden pelaajien yhteysosoitteet. Tämän jälkeen vertaisverkon muut tahot voivat aloittaa yhteydenmuodostuksen tämän uuden tahon kanssa. On huomioitavaa, ettei uutta pelaajaa voi päästää pelaamaan, ennenkuin kaikki vertaisverkon yhteydet on saatu luotua onnistuneesti. Lisäksi kehittäjien on varauduttava tilanteisiin, joissa vertaisverkon ylläpitäjäksi valittu taho päättääkin katkaista yhteyden ensimmäisenä.

9. YHTEENVETO

Tämä diplomityö käsitteli erään laajasti käytössä olevan reaaliaikaisen moninpelirajapinnan toimintaperiaatteita ja Hyperkani Oy -yrityksen kannalta rajapinnan käyttöön liittyviä suurimpia epäkohtia sekä ongelmia. Vastaukseksi näihin ongelmiin esitettiin vaihtoehtoista ja järjestelmäriippumatonta ratkaisumallia, jossa kyseiset ongelmat on ratkaistu tai kierretty muulla tavoin. Ratkaisumallista toteutettiin myös prototyyppi ratkaisun käyttökelpoisuuden ja toiminnan osoittamiseksi.

Ratkaisumalli, reaaliaikaisen moninpelirajapinnan prototyyppi, osoittautui toimivaksi sekä helposti laajennettavaksi komponentiksi. Toistaiseksi sen sisältämät toiminnot jäivät suppeiksi ja osoittavat ainoastaan ratkaisun toimivuuden, mutta sen lähdekoodien ollessa yrityksen hallinnassa, komponentin kehittäminen sekä muokkaaminen yrityksen toiveiden mukaan on mahdollista ilman erillisiä rajoitteita. Oleellisimmat puutteelliset toiminnot on myös dokumentoitu ja näille puutteille on esitetty alustavat toimintaperiaatteet niiden toteuttamista varten.

Reaaliaikaisen moninpelirajapinnan prototyypin suorituskykykä mitattiin viiveiden käyttäytymisellä pakettien koon tai lähetysvälin muuttuessa. Tällainen toiminta on tavallista verkkopeleissä, joten yrityksen kannalta on oleellista tietää, miten järjestelmä käyttäytyy edellämainituissa tilanteissa. Havaittiin, että järjestelmä toimii tehokkaimmin pakettien koon pysyessä ethernet-kehyksen sallimissa rajoissa ja pakettien lähetysvälin pysyessä vähintään 50ms suuruisina. Suuremmat paketit sekä pienemmät lähetysvälit toimivat edelleen luotettavasti, mutta saattavat vaikuttaa pakettien välisiin viiveisiin. Nämä tulokset eivät juurikaan eroa aiemmin käytössä olleen moninpelirajapinnan rajoituksista, joten prototyypin toimintaa voidaan pitää riittävän tehokkaana.

Ratkaisuksi esitettyä prototyyppiä ei kuitenkaan otettu yrityksessä käyttöön, sillä testien perusteella havaittiin, ettei vertaisverkon luominen kahden mobiililaitteen välille onnistu riittävän usein ilman relay-palvelinta. Tämä johtuu siitä, että useimmat mobiiliverkkoja tarjoavat operaattorit hyödyntävät osoitteenmuunnoksia sekä palomuureja tarjoamissaan mobiililiittymissä, jolloin kumpikaan vertaisverkon taho ei pysty muodostamaan suoraa yhteyttä laitteiden välille. Välityspalvelimen kautta

yhteys saadaan onnistuneesti muodostettua viiveiden kasvamisen kustannuksella.

Prototyyppi osoittaa kuitenkin ratkaisun olevan teknisesti mahdollinen ja osoitteenmuunnosten muodostama ongelma voi mahdollisesti poistua IPv6-osoiteavaruuden laajamittaisen käyttöönoton yhteydessä. Tässä IP-protokollan päivitetyssä versiossa erilaisia verkko-osoitteita on tarjolla moninkertainen määrä nykyisin käytössä olevaan IPv4-osoiteavaruuteen verrattuna, jolloin osoitteenmuunnos voi mahdollisesti jäädä tarpeettomaksi myös mobiililaitteille. Palomuurien muodostamaan ongelmaan tämä ei tietenkään vaikuta. Ongelmista johtuen työssä on myös pohdittu erilaisia tapoja yhteyden muodostamiseksi osoitteenmuunnosten takana olevien laitteiden välille.

Vertaisverkkojen hyödyntäminen eri käyttötarkoituksiin yleistyy kuitenkin jatkuvasti. Tulevaisuudessa onkin mahdollista, että esimerkiksi operaattorit voivat mainostaa tarjoavansa osoitteenmuunnoksetonta yhteyttä asiakkailleen eräänlaisena kilpailuvalttina. Vertaisverkot poistavat keskitettyjen palvelinten muodostaman ongelman viivekriittisissä moninpeleissä sillä kustannuksella, että pelien tilatiedot hajautetaan verkon eri tahojen kesken. Tämä monimutkaistaa asiakasohjelmien kehittämistä, mutta sen tarjoamat hyödyt ovat erityisesti reaaliaikakriittisissä sovelluksissa ylivertaisia asiakas-palvelin-arkkitehtuuriin nähden.

LÄHTEET

[1] Android Developers. Application Fundamentals. [WWW]. [Viitattu 13.10.2015]. Saatavissa: http://developer.android.com/guide/components/fundamentals.html

[2] G.J. Armitage. An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3. 11th IEEE International Conference on Networks (ICON 2003). 2003. 5 sivua.

[3] S. Agarwal, J.R. Lorch. Matchmaking for Online Games and Other Latency-Sensitive P2P Systems. Conference of the Special Interest Group on Data Communications. 2009. August 17-21, Barcelona, Spain. 12 sivua.

[4] A. Anitha, J. JayaKumari, G.V. Mini. A Survey of P2P Overlays in Various Networks. Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies. 2011. 5 sivua.

[5] A. Beznosyk, P. Quax, K. Coninx, W. Lamotte. Influence of Network Delay and Jitter on Cooperation in Multiplayer Games. Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry. 2011. 4 sivua.

[6] Bungie. Halo 2 Matchmaking Overview. 2010. [WWW]. [Viitattu 12.3.2015]. Saatavissa: http://halo.bungie.net/stats/content.aspx?link=h2matchmaking

[7] M. Claypool, K. Claypool. On Latency and Player Actions in Online Games. Worcester Polytechnic Institute, DigitalCommons. 2006. 14 sivua.

[8] Cocos2D-X. Cocox2D-X Developers Manual. 2015. [WWW]. [Viitattu 21.10.2015]. Saatavissa: http://www.cocos2d-x.org/wiki/Cocos2d-x

[9] L. D'Acunto, J.A. Pouwelse, H.J. Sips. A Measurement of NAT & Firewall Characteristics in Peer to Peer Systems. ResearchGate, Article, January 2009. 6 sivua.

[10] M. Dick, O. Wellnitz, L. Wolf. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. Conference NetGames'05, 4th Workshop on Network and System Support For Games. 2005. October 10-11, Hawthorne, NY, USA. 7 sivua.

[11] S. Douglas, E. Tanin, A. Harwood, S. Karunasekera. Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture. Proceedings of the

International Conference on Information and Automation, December 2005, Colombo, Sri Lanka. 6 sivua.

[12] J. Downling, J. Sacha, S. Haridi. Improving ICE Service Selection in a P2P System using the Gradient Topology. IEEE International Conference on Self-Adaptive and Self-Organizing Systems. 2007. 5 sivua.

[13] J. Edwards. Android Market Share. 2014. [WWW]. [Viitattu 23.8.2015]. Saatavissa: http://www.businessinsider.com/iphone-v-android-market-share-2014-5?IR=T

[14] K. Egevang. The IP Network Address Translator (NAT). Network Working Group. Request For Comments 1631. 1994. 10 sivua.

[15] E. Exist, C. Hornig. A Standard for the Transmission of IP Datagrams over Ethernet Networks. Internet Engineering Task Force (IETF). Request For Comments 894. 1984. 3 sivua.

[16] L. Fan, P. Trinder, H. Taylor. Design Issues for Peer-to-Peer Massively Multiplayer Online Games. School of Mathematical and Computer Sciences, Herot-Watt University, Edinburgh, UK. Inderscience Enterprises Ltd. 2009. 11 sivua.

[17] B. Ford, P. Srisuresh, D. Kegel. Peer-to-Peer Communication Across Network Address Translators. 2005. [WWW]. [Viitattu 14.10.2015]. Saatavissa: https://www.usenix.org/legacy/event/usenix05/tech/general/full_papers/ford/ford_html/

[18] N. Gandhewar, R. Sheikh. Google Android: An Emerging Software Platform for Mobile Devices. International Journal on Computer Science and Engineering (IJCSE). 2010. 6 sivua.

[19] Google Developers. About libjingle. 2012. [WWW]. [Viitattu 24.3.2015]. Saatavissa: https://developers.google.com/talk/libjingle/index

[20] Google Developers. Creating a libjingle Application. 2012. [WWW]. [Viitattu 24.3.2015]. Saatavissa: https://developers.google.com/talk/libjingle/building_an_app

[21] Google Developers. FAQ: Open Communications. [WWW]. [Viitattu 19.10.2015]. Saatavissa: https://developers.google.com/talk/open_communications

[22] Google Developers. Google Play Games Services. 2014. [WWW]. [Viitattu 18.3.2015]. Saatavissa: https://developers.google.com/games/services/

[23] Google Developers. Get Started with Play Games Services for Web. [WWW]. [Viitattu 2.10.2015]. Saatavissa: https://developers.google.com/games/services/web/gettingstarted

[24] Google Developers. Google Play Games Services Real-time Multiplayer Concepts. 2014. [WWW]. [Viitattu 18.3.2015]. Saatavissa: https://developers.google.com/games/services/common/concepts/realtimeMultiplayer

[25] Google Developers. Google Play Games Services SDK Downloads. [WWW]. [Viitattu 2.10.2015]. Saatavissa: https://developers.google.com/games/services/downloads/sdks

[26] Google Developers. Introduction to libjingle. [WWW]. [Viitattu 18.10.2015]. Saatavissa: https://developers.google.com/talk/libjingle/developer_guide

[27] Google Play. Bomber Friends. [WWW]. [Viitattu 11.10.2015]. Saatavissa: https://play.google.com/store/apps/details?id=com.hyperkani.bomberfriends

[28] Google Transparency Report. Known disruptions of traffic to Google products and services. [WWW]. [Viitattu 18.10.2015]. Saatavissa: http://www.google.com/transparencyreport/traffic/disruptions/

[29] I. Grigorik. High Performance Browser Networking. 1. painos. 2013. O'reilly. 382 sivua.

[30] P. Hancke. WebRTC and XMPP. XMPP Summit 2013, XMPP Standards Foundation Developer Conference. Estos, Communications Solutions. Luento. 17 sivua.

[31] D.A. Heger. Mobile Devices - An Introduction to the Android Operating Environment. Design, Architecture, and Performance Implications. DHTechnologies. 2011. 7 sivua.

[32] E. Howard, C. Cooper, M.P. Wittie, S. Swinford, Q. Yang. Cascading Impact of Lag on User Experience in Multiplayer Games. USENIX Symposium on Networked Systems Design and Implementation. 2015. May 4-6, Oakland, CA. 2 sivua.

[33] P. Kabus, A.P. Buchmann. Design of a Cheat-Resistant P2P Online Gaming System. Databases and Distributed Systems Group. Technische Universitaet Darmstadt, Germany. 2007. 8 sivua.

[34] B. Knutsson, H. Lu, W. Xu, B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. Department of Computer and Information Science, University of Pennsylvania. IEEE International Conference on Computer Communications. 2004. 12 sivua.

[35] Libnice. Libnice Reference Manual. [WWW]. [Viitattu 21.10.2015]. Saatavissa: http://nice.freedesktop.org/libnice/

[36] Libnice. The GLib ICE implementation. [WWW]. [Viitattu 19.10.2015]. Saatavissa: http://nice.freedesktop.org/wiki/

[37] Libnice. The nice Archives. [WWW]. [Viitattu 20.10.2015]. Saatavissa: http://lists.freedesktop.org/archives/nice/

[38] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim. A Survey And Comparison Of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys & Tutorials. January 2005, Volume 7, No 2. 22 sivua.

[39] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, J. Hildebrand. XEP-0166: Jingle. Specification. XMPP Standards Foundation. 2009.

[40] V. Matos. Lesson 3, Application's Life Cycle. Cleveland State University. 2014. 45 sivua.

[41] D.J. McCaffery, J. Finney. The Need for Real Time Consistency Management in P2P Mobile Gaming Environments. International Conference on Advances in Computer Entertainment. 2004. June 3-5, Singapore. 9 sivua.

[42] L. Mäkinen, J.K. Nurminen. Measurements on the Feasibility of TCP NAT Traversal in Cellular Networks. Helsinki University of Technology, Laboratory of Software Technology. Nokia Research Center. 2008. 7 sivua.

[43] C. Neumann, N. Prigent, M. Varvello, K. Suh. Challenges in Peer-to-Peer Gaming. ACM SIGCOMM Computer Communication Review. Vol. 37. Number 1. January 2007. 4 sivua.

[44] L. Pantel, L. C. Wolf. On the Impact of Delay on Real-Time Multiplayer Games. NOSSDAV '02, Proceedings of the 12 international workshop on Network and operating systems support for digital audio and video. 2002. May 12-14, Miami, Florida, USA. 7 sivua.

[45] J. Postel. Transmission Control Protocol. Information Sciences Institute. Request For Comments 793. 1981. 85 sivua.

[46] J. Postel. User Datagram Protocol. Information Sciences Institute. Request For Comments 768. 1980. 3 sivua.

[47] M. Poulin, L.R. Maldague, A. Daigle, F. Gagnon. NAT Traversal in Peer-to-Peer Architecture. Technical Report SCE-12-04. 2012. 19 sivua.

[48] R. Rogers. Learning Android Game Programming. 1. painos. 2011. Pearson. 444 sivua.

[49] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Internet Engineering Task Force (IETF). Request For Comments 5245. 2010. 116 sivua.

[50] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy. STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). Request for Comments 3489. 2003. 47 sivua.

[51] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. 2004. Jabber Software Foundation. Request for comments 3920. 2004. 90 sivua.

[52] R. Schollmeier. A Definition of Peer-toPeer Networking for the Classification of Peer-to-Peer Architectures and Applications. P2P '01 Proceedings of the First International Conference on Peer-to-Peer Computing. 2001. 2 sivua.

[53] V. Silva. Pro Android Games. 1. painos. 2009. Apress. 298 sivua.

[54] Statista, The Statistics Portal. Mobile gaming revenue worldwide from 2008 to 2017. 2014. [WWW]. [Viitattu 25.8.2015]. Saatavissa rajoitetusti: http://www.statista.com/statistics/260157/mobile-gaming-revenue-worldwide/

[55] Statista, The Statistics Portal. Number of available applications in the Google Play Store from December 2009 to July 2015. 2015. [WWW]. [Viitattu 21.8.2015]. Saatavissa: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[56] Statista, The Statistics Portal. Number of smartphone users worldwide from 2012 to 2018. 2014. [WWW]. [Viitattu 25.8.2015]. Saatavissa: http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[57] B. Sterman, D. Schwartz. NAT Traversal in SIP. Deltathree, The IP Communications Network. 2013. 16 sivua.

[58] K. Systä. Hajautettujen Järjestelmien Perusteet - Arkkitehtuurit 2/2. 2013. Tampereen Teknillinen Yliopisto. [WWW]. [Viitattu 10.3.2015]. Luentokalvot saatavissa: http://www.cs.tut.fi/~systa/hajap/03arkkitehtLoppuun2013.pdf

[59] K. Systä. Hajautettujen Järjestelmien Perusteet - Kommunikointi. 2013. Tampereen Teknillinen Yliopisto. [WWW]. [Viitattu 12.10.2015]. Luentokalvot saatavissa: http://www.cs.tut.fi/~systa/hajap/07kommunikoinnista2013.pdf

[60] Unity documentation. Networking overview. 2015. [WWW]. [Viitattu 15.10.2015]. Saatavissa: http://docs.unity3d.com/Manual/UNetOverview.html

[61] M. Vitas. ART vs Dalvik - introducing the new Android runtime in KitKat. Technology, Learning, Android Development. December 4th, 2013.

[62] A. Voltornist. Mobile broadband reach expanding globally. 2014. [WWW]. [Viitattu 15.10.2015]. Saatavissa: https://gsmaintelligence.com/research/2014/12/mobile-broadband-reach-expanding-globally/453/

[63] Z. Wang, Z. Qian, Q. Xu, Z. Morley Mao, M. Zhang. An Untold Story of Middleboxes in Cellular Networks. SIGCOMM'11, Special Interest Group on Data Communication (SIGCOMM) on the applications, technologies, architectures, and protocols for computer communication. 2011. August 15-19, Toronto, Ontario, Canada. 13 sivua.

[64] Wireshark Foundation. Wireshark, network protocol analyzer. [WWW]. [Viitattu 11.10.2015]. Saatavissa: https://www.wireshark.org/

[65] M. Zechner, R. Green. Beginning Android Games. 1. painos. 2011. Apress. 686 sivua.